Introduction and Objectives

Welcome to the Develop Scope and Approach lesson. After completing this lesson, you will understand the first of the five major steps of developing a software cost estimate.

Lesson Objectives

- Consider the purpose of a software estimate and its impact on the development of the estimate.
- Evaluate the appropriateness and completeness of the estimating plan for a software estimate.
- Summarize the key components of the baseline for a software-intensive system given program documentation such as a Cost Analysis Requirements Description (CARD).
- Appraise the adequacy of the cost element structure (CES) for a software estimate.
- Develop a set of ground rules and assumptions (GR&A) for a software estimate.

Note that throughout "software estimate" is referred to as cost estimate. While the primary focus is the cost estimate, it is important to keep in mind that the sizing estimate and the schedule estimate are also crucial, and that the three should be developed in conjunction with each other.

D

Long Description

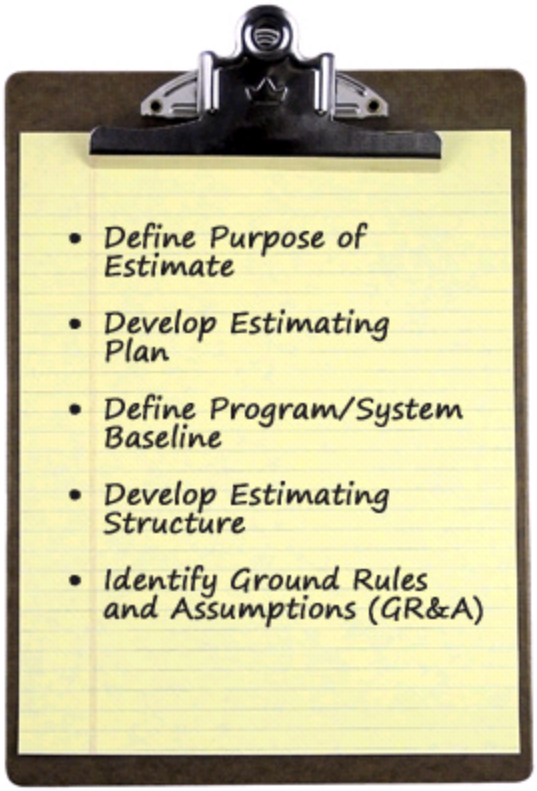
Graphic illustrates the steps of the Cost Estimating process. The steps from left to right are:
Develop Scope and Approach (highlighted) Collect and Analyze Data, Develop Estimate Methodology, Consider Risk and Uncertainty, and Document and Present Estimate.

Sub-steps

Developing the Scope and Approach is composed of the following five sub-steps:

- Define Purpose of Estimate
- Develop Estimating Plan
- Define Program/System Baseline
- Develop Estimating Structure
- Identify Ground Rules and Assumptions (GR&A)

This lesson addresses each of the sub-steps in greater detail.



D

Long Description

Clipboard listing the five sub-steps.

- Define Purpose of Estimate
- Develop Estimating Plan
- Define Program/System Baseline
- Develop Estimating Structure
- Identify Ground Rules and Assumptions (GR&A)

Importance of Software

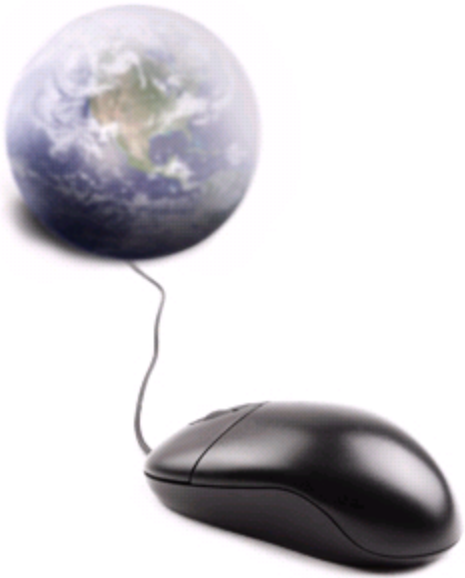
While software cost estimating draws from the standard set of cost estimating processes, methodologies, analytical approaches, and tools, it merits special consideration for several reasons. In particular, software is:

	Intangible	Risky	Expensive	Rapidly Changing

Ubiquitous

Software is everywhere. It is hard to find systems these days without significant software content.

This brings with it both the challenge of getting software estimates right and the opportunity to a collect a large data set with potential applicability across a wide range of commodities.



Popup Text

Ubiquitous

Software is everywhere. It is hard to find systems these days without significant software content.

This brings with it both the challenge of getting software estimates right and the opportunity to a collect a large data set with potential applicability across a wide range of commodities.

Intangible

Software is not a physical object like a plane or ship or tank, but rather the proverbial collection of 0s and 1s, and can therefore be hard to define and decompose.

Because well-written software can be very powerful and deceptively simple, there is the peril of being misled into believing that software can give you something for nothing (or at least for very little).

Risky

On the contrary, large software development projects are very complex, and are prey to diseconomies of scale and regression to the mean.

Ill-defined or ever-changing requirements, parallel development, code growth, reuse optimism, over-reliance on COTS, communication breakdown, poor interface design, and lack of familiarity with development tools are some of the risks that can befall a software project.

Popup Text

Expensive

Department of Defense (DoD) spending for software intensive systems is significant and continues to increase.

For example, DoD spending just on software embedded in weapon systems is estimated to be between $24 billion to $32 billion annually.

Popup Text

Rapidly Changing

Information Technology (IT) in general and software in particular are in continual flux, which necessitates care in applying historical data to estimating future efforts (though not abandonment of historical data altogether).

Information Technology (IT)

Any equipment or interconnected system or subsystem of equipment that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information by the executive agency. IT includes computers, ancillary equipment, software, firmware and similar procedures, services (including support services), and related resources, including National Security Systems (NSS). It does not include any equipment that is acquired by a federal contractor incidental to a federal contract.

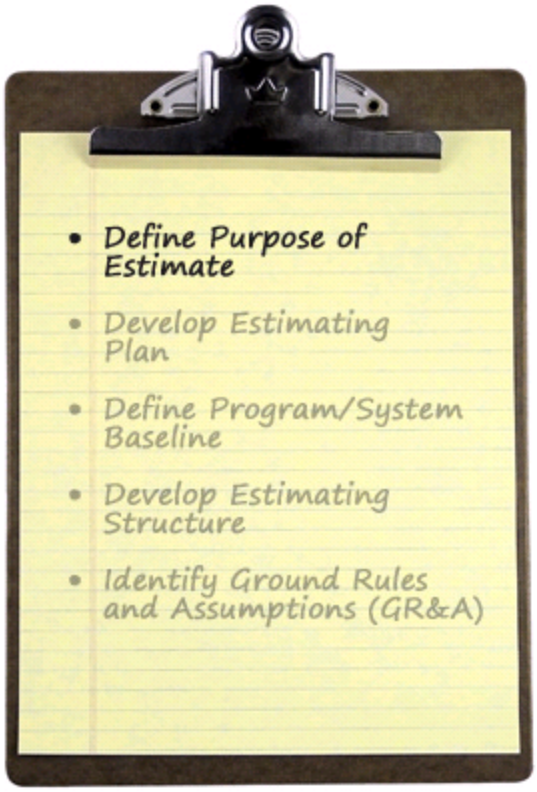
Step 1A: Define Purpose of Estimates

The purpose of an estimate may affect everything from the content, level of detail, and timeframe covered by the estimate itself, to the team preparing the estimate and the timeline and constraints they are operating under.

Where public law or DoD policy and procedures are involved, specific guidelines are given.

The ideal cost estimate is both comprehensive and holistic, as embodied by a life cycle cost estimate (LCCE), and unbiased and detached from program advocacy, as embodied by an independent cost estimate (ICE).

While these ideals should not be compromised, different purposes will drive the tailoring of the estimate in different ways.



Popup Text

Life Cycle Cost Estimate (LCCE)

A cost estimate that covers all of the costs projected for a system's life cycle, and which aids in the selection of a cost-effective total system design, by comparing costs of various trade-offs among design and support factors to determine their impact on total system acquisition and ownership costs.

Independent Cost Estimate (ICE)

A Life Cycle Cost Estimate (LCCE) for Acquisition Category (ACAT) I or IA programs prepared by an office or other entity that is not under the supervision, direction, or control of the military department, defense agency, or other component of DoD that is directly responsible for carrying out the development or acquisition of the program, or if the decision authority has been delegated to a component, prepared by an office or other entity that is not directly responsible for carrying on the development or acquisition of the program.

Step 1A: Define Purpose of Estimates, Cont.

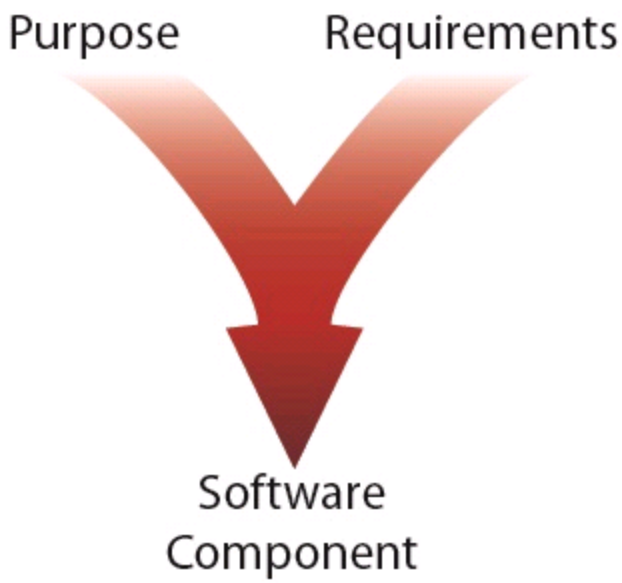
It is important to note that a software estimate is rarely "stand-alone" but rather is usually part of a larger estimating effort.

The purpose and requirements for the estimate generally "flow down" to the software component. Where this component is particularly critical, the program is often said to involve a "software-intensive" system.

There are three major types of cost estimates, driven by different purposes:

- Budgetary Estimates
- Comparative Estimates
- Planning Estimates

Before discussing each cost estimate in greater detail, it is important to review the two major programs within the DoD, Major Automated Information Systems (MAIS) and Major Defense Acquisition Programs (MDAPs).



Questions Managers Should Ask

Popup Text

Questions Managers Should Ask

Are the objectives of the estimate clear and correct?

- Are the objectives of the estimate stated in writing?
- Is the life cycle to which the estimate applies clearly defined?

Step 1A: Define Purpose of Estimates - MAIS vs. MDAPs

The largest and most important programs are designated as Acquisition Category I, or ACAT I, and are divided into:

- Major Automated Information Systems (MAIS) – business systems designated ACAT IA
- Major Defense Acquisition Programs (MDAPs) – weapon systems designated ACAT ID or IC.

In some cases, MAIS programs are large or important enough that they are also designated as MDAPs. By their nature, MAIS programs are almost always software-intensive, but with software such a critical part of modern technology, most MDAPs are software-intensive as well. As each of the three types of estimates is discussed, particular requirements for MDAPs and MAIS will be noted.

Step 1A: Define Purpose of Estimates - MAIS vs. MDAPs, Legal Requirements

Title 10, United States Code, Section 2434, requires that the Secretary of Defense consider an independent estimate of the life-cycle cost an MDAP prior to granting Milestone B or Milestone C approval.

The ICE is a full LCCE, and it must be produced by an entity outside the development and acquisition chain(s) of command. This ICE falls into the category of Budgetary Estimates.

Title 44, United States Code, Section 3506 and the Clinger-Cohen Act of 1996, Section 5122 require that a cost-benefit analysis (CBA) be performed for all MAIS acquisitions.

Furthermore, DoD Instruction 5000.2, Enclosure 3, requires that an Economic Analysis (EA) be performed in support of the Milestone A, Milestone B, and Full-Rate Production (FRP) decision reviews for MAIS. The CBA and EA fall into the category of Comparative Estimates.

Popup Text

Milestone

The point at which a recommendation is made and approval sought regarding starting or continuing an acquisition program, i.e., proceeding to the next phase. Milestones established by DoDI 5000.02 are: Milestone A that approves entry into the Technology Development (TD) phase; Milestone B that approves entry into the Engineering and Manufacturing Development (EMD) phase; and Milestone C that approves entry into the Production and Deployment (P&D) phase.

Cost Benefit Analysis (CBA)

An analytic technique that compares the costs and benefits of investments, programs, or policy actions in order to determine which alternative or alternatives maximize net profits. Net benefits of an alternative are determined by subtracting the present value of costs from the present value of benefits.

Economic Analysis (EA)

A systematic approach to selecting the most efficient and cost-effective strategy for satisfying an agency's need. An EA evaluates the relative worth of different technical alternatives, design solutions, and/or acquisition strategies, and provides the means for identifying and documenting the costs and associated benefits of each alternative to determine the most cost-effective solution. Normally associated with Automated Information System (AIS) acquisition programs.

Full-Rate Production (FRP)

Contracting for economic production quantities following stabilization of the system design and validation of the production process.

Step 1A: Define Purpose of Estimates - Budgetary Estimates

The purpose of Budgetary Estimates is to support the allocation of ample funding in the proper program elements (PEs) to enable program success.

The results of a Budgetary Estimate are used to establish, compare to, and update program budgets and targets (thresholds and objectives).

The comparisons are used to assess affordability, and if unfavorable, may lead to program restructuring or cancelation.

Popup Text

Program Elements (PEs)

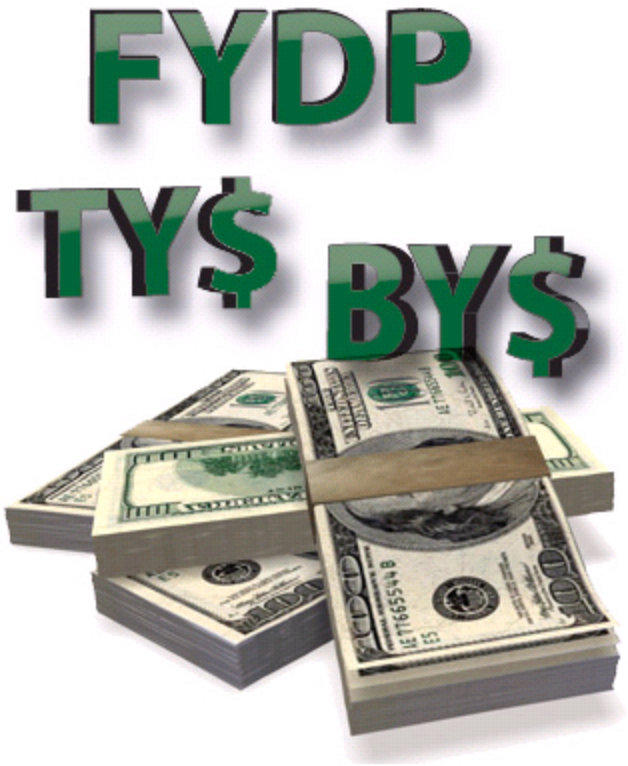
The basic building block of the 11 major programs of the Future Years Defense Program (FYDP). It is "an integrated combination of men, equipment, and facilities, which together constitute an identifiable military capability or support activity." It also identifies the mission to be undertaken and the organizational entities to perform the mission. Elements may consist of forces, manpower, materials, services, and/or associated costs as applicable. A PE consists of seven digits ending with a letter indicating the appropriate Service.

Step 1A: Define Purpose of Estimates - Budgetary Estimates, Cont.

A Budgetary Estimate may focus on a subset of life cycle cost (LCC), often restricted in time – to the span of years reflected in the Future Years Defense Program (FYDP), say – and mapped to funding lines. It is primarily in Then-Year dollars (TY$) for budgeting purposes, though comparison with targets is often done in Base Year dollars (BY$). It should reflect exactly the scope of the program of record.

For software, this means the current architecture, requirements allocation, development increments, and other components of baseline definition which is addressed in Step 1C. Keep in mind that early on – pre-contract award, for example – there may be some significant uncertainties, such as which contractor will be performing software development, that should be reflected in the estimate.

The most common example of Budgetary Estimates are those associated with the Program Objectives Memorandum (POM) submission, part of the Planning, Programming, Budgeting, and Execution (PPBE) Process, but the estimates required for MDAPs and MAIS programs for milestone reviews as part of the Acquisition process are essentially budgetary in nature. Estimates are also required for certification when Nunn-McCurdy breaches occur.



Popup Text

Life Cycle Cost (LCC)

For a defense acquisition program, LCC consists of research and development (R&D) costs, investment costs, operating and support costs, and disposal costs over the entire life cycle. These costs include not only the direct costs of the acquisition program, but also include indirect costs that would be logically attributed to the program. In this way, all costs that are logically attributed to the program are included, regardless of funding source or management control.

Future Years Defense Program (FYDP)

A DoD database and internal accounting system that summarizes forces and resources associated with programs approved by the Secretary of Defense (SECDEF). Its three parts are the organizations affected, appropriations accounts (e.g., research, development, test, and evaluation (RDT&E); operation and maintenance (O&M); etc.) and the 11 major force programs (e.g., strategic forces, mobility forces, research and development (R&D), etc.). The FYDP allows a "crosswalk" between DoD's internal system of accounting via 11 major force programs and congressional appropriations. The primary data element in the FYDP is the program element (PE). The FYDP is updated twice during the Planning, Programming, Budgeting and Execution (PPBE) process cycle: submission of the concurrent Program Objectives Memorandum (POM)/Budget Estimate Submission (BES) (usually July/August), and submission of the President's Budget (PB) (early February the year following).

Current Year (CY) Dollars, Then-Year (TY) Dollars

Dollars that include the effects of inflation or escalation and/or reflect the price levels expected to prevail during the year at issue.

Popup Text

Base Year (BY)

Reference period that determines a fixed price level for comparison in economic escalation calculations and cost estimates. The price level index for the BY is 1.000.

Program Objectives Memorandum (POM)

The final product of the programming process within DoD, a Component's POM displays the resource allocation decisions of the military department in response to, and in accordance with the Defense Planning and Programming Guidance (DPPG). The POM shows programmed needs 5 years hence (e.g., in FY 2012, POM 2014–2018 was submitted).

Planning, Programming, Budgeting, and Execution (PPBE) Process

The primary Resource Allocation Process (RAP) of DoD. It is one of three major decision support systems for defense acquisition along with Joint Capabilities Integration and Development System (JCIDS) and the Defense Acquisition System. It is a formal, systematic structure for making decisions on policy, strategy, and the development of forces and capabilities to accomplish anticipated missions. PPBE is an annual process which produces the Secretary's Defense Planning and Programming Guidance (DPPG), five year approved Program Objectives Memoranda (POMs), and one year Budget Estimate Submissions (BES) for the military departments and defense agencies, and the DoD portion of the President's Budget (PB).

Nunn-McCurdy Breach

Refers to *Title 10, U.S.C. § 2433, Unit Cost Reports (UCRs)*. This amendment to Title 10 was introduced by Senator Sam Nunn and Congressman Dave McCurdy in the National Defense Authorization Act (NDAA) for Fiscal Year (FY) 1982. Requires that Acquisition Category I (ACAT I) program managers (PMs) maintain current estimates of Program Acquisition Unit Cost (PAUC) and Average Procurement Unit Cost (APUC). If the PAUC or APUC increases by 25 percent or more over the current Acquisition Program Baseline (APB) objective, or 50 percent or more over the original APB objective, the program must be terminated unless the Secretary of Defense (SECDEF) certifies to Congress that the program is essential to national security.

Step 1A: Define Purpose of Estimates - Comparative Estimates

Comparative Estimates support the decision among multiple courses of action, each of which may have different costs and performance characteristics.

They most reflect the ideal of the comprehensive LCCE, taking into account all relevant future costs related to the decision at hand, even if not directly attributable to the program. Often, they require significant adjustments to certain alternatives to make them comparable.

By definition, at most one of the alternatives can reflect the current program of record. Comparative Estimates for software may consider alternatives such as centralized vs. distributed architecture, developed software vs. commercial off-the-shelf (COTS), and organic vs. contractor support.

D

Long Description

Image of tug-of-war with the following text:

Central vs. Distributed
Developed vs. COTS
Organic vs. Contractor

Step 1A: Define Purpose of Estimates - Comparative Estimates, Cont.

The purpose of the aforementioned required EA is to determine the best automated information system (AIS) program acquisition alternative, by assessing the net costs and benefits of the proposed AIS program relative to the status quo.

In general, the best alternative will be the one that meets validated capability needs at the lowest lifecycle cost (measured in present value terms), and/or provides the most favorable return on investment. Whenever an EA is required for a MAIS program, a Component Cost Estimate (CCE) is also required.

Because Comparative Estimates often entail a trade-off between performance and costs, their greatest affinity is with the Joint Capabilities Integration and Development System (JCIDS), but they are also common as part of the Acquisition process as described above.

Popup Text

Component Cost Estimate (CCE)

The generic term “DoD Component Cost Estimate” is used to provide considerable latitude to each military service or defense agency as to the actual responsibility for this cost estimate. In some cases, a military service assigns the responsibility to the program office (PO), which then provides a PO Life-Cycle Cost Estimate (PLCCE). In other cases, the DoD component may adopt a more corporate approach in which an initial program office (PO) cost estimate is subject to considerable review and possible adjustment as determined by the Service Cost Center or defense agency equivalent.

Joint Capabilities Integration and Development System (JCIDS)

Supports the Chairman of the Joint Chiefs of Staff (CJCS) and the Joint Requirements Oversight Council (JROC) in identifying, assessing, and prioritizing joint military capability needs as required by law. The capabilities are identified by analyzing what is required across all joint capability areas to accomplish the mission.

Step 1A: Define Purpose of Estimates - Planning Estimates

Planning Estimates support Acquisition directly by enabling the establishment and successful execution of contracts for needed goods and services.

Their scope is usually limited to the contract(s) under consideration and costs to be incurred by the prime contractor and all the tiers of subcontractors.

Planning Estimates include the Independent Government Cost Estimate (IGCE) associated with a Request For Proposal (RFP); the basis of estimate (BOE) submitted by the respondents to justify their cost proposals, which may be adjusted by the government as part of its cost analysis; and the estimates used to establish the performance measurement baseline (PMB) to support execution of earned value management (EVM) by the contractor.

Planning Estimates, especially of the last two varieties, are often the most detailed and expensive to produce.

Popup Text

Independent Government Cost Estimate (IGCE)

An estimate of the cost for goods and/or estimate of services to be procured by contract. Such estimates are prepared by government personnel, i.e., independent of contractors.

Request for Proposal (RFP)

A solicitation used in negotiated acquisition to communicate government requirements to prospective contractor and to solicit proposals.

Basis of Estimate (BOE)

A document describing the resources required for a proposed effort and the associated methodology used to estimate those resources and accompanying rationale.

Performance Measurement Baseline (PMB)

The Performance Measurement Baseline (PMB) is a time-phased budget plan for accomplishing work, against which contract performance is measured. It includes the budgets assigned to scheduled control accounts and the applicable indirect budgets. For future effort, not planned to the control account level, the PMB also includes budgets assigned to higher level Contractor Work Breakdown Structure (CWBS) elements, and to undistributed budgets. It does not include management reserve.

Step 1A: Define Purpose of Estimates - Planning Estimates, Cont.

For software, Planning Estimates should reflect the specific prime contractor and subcontractors performing software development, taking into account their productivities and rates; the agreed-upon bill of materials (BOM), including any development infrastructure and COTS software licenses; and specific aspects of the technical proposal/plan, such as development approach, staffing plan, test activities, etc.

Because software development is labor-intensive, it is important that all supporting activities are accounted for and properly time-phased, and that reasonable earned value metrics are chosen to accurately reflect progress.

Popup Text

Bill of Materials (BOM)

A descriptive and quantitative listing of all of the materials, supplies, parts, and components required to produce a complete end item of material, assembly, or subassembly, to overhaul or repair such an item, or to construct or repair a structure or facility.

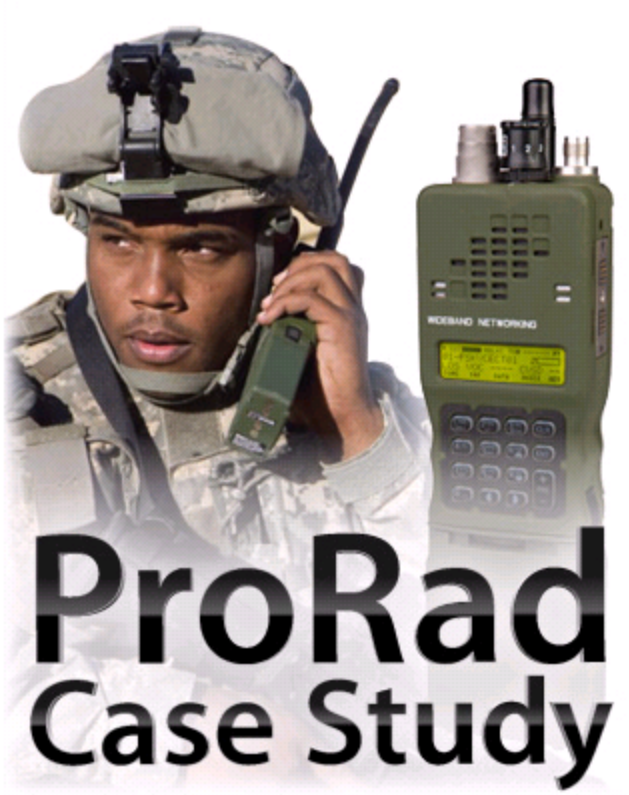
Step 1A: Define Purpose of Estimates - ProRad Estimate Purpose

In the late 1990s, the DoD conducted a capabilities-based assessment that resulted in a material need for a software-defined radio (SDR). The program was later designated as the Joint Programmable Radio (Joint ProRad), ACAT ID.

In the scenario, the ProRad program underwent a Milestone B (MS B) review to approve entry into Phase B Engineering and Manufacturing Development (EMD). As part of this process, the Office of the Secretary of Defense (OSD) Cost Analysis Improvement Group (CAIG) conducted an independent cost estimate (ICE) on ProRad, since it is an ACAT ID program.

The cost center from the acquiring service also coordinated the establishment of a component cost position. The software estimating considerations germane to both estimates are addressed throughout this module.

The CAIG has since been subsumed into the Cost Assessment and Program Evaluation (CAPE) organization, but because this is a historical example, CAIG will continue to be used.



Popup Text

Engineering and Manufacturing Development (EMD)

The third phase of the life cycle as defined and established by DoDI 5000.02. This phase consists of two efforts—Integrated System Design (ISD) and System Capability and Manufacturing Process Demonstration (SC&MPD)—and begins after Milestone B. It also contains a Post-Critical Design Review (CDR) Assessment at the conclusion of the ISD effort and may also contain a Post Preliminary Design Review (PDR) Assessment early in ISD for non-major defense acquisition programs. A program planning to proceed into SC&MPD at the conclusion of ISD will first undergo a Post-CDR Assessment to confirm design maturity and the initial product baseline.

Cost Analysis Improvement Group (CAIG)

The CAIG function and personnel were transferred to the Office of the Secretary of Defense, Director for Cost Assessment and Program Evaluation (D,CAPE) by the Weapon Systems Acquisition Reform Act (WSARA) of 2009. Within the office of D, CAPE, the Deputy Director for Cost Assessment oversees policy and procedures for cost estimating and conducts independent cost estimates for major defense acquisition programs (MDAPs) and major automated information systems (MAIS).

Component Cost Position

At milestone reviews, each DoD component establishes a DoD component-level cost position for its Major Defense Acquisition Programs (MDAPs). To support DoD's full funding policy for acquisition programs, as well as specific statutory certifications and regulatory requirements, the DoD component is expected to fully fund the program to this cost position in the current President’s Budget (PB)/Future Years Defense Program (FYDP), or commit to full funding of the cost position in the next PB/FYDP, with identification of specific offsets to address any funding shortfalls that may exist in the current FYDP.

Popup Text

Cost Assessment and Program Evaluation (CAPE)

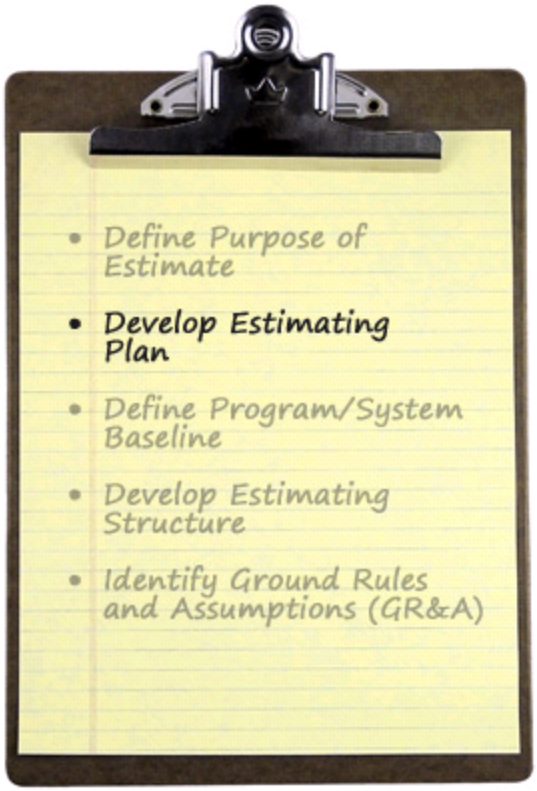
The Director, CAPE is the principal staff assistant to the Secretary of Defense for Cost Assessment and Program Evaluation. The Director's principal responsibilities include:

- Analyze and evaluate plans, programs, and budgets in relation to U.S. defense objectives, projected threats, allied contributions, estimated costs, and resource constraints.
- Review, analyze, and evaluate programs, including classified programs, for executing approved policies.
- Provide leadership in developing and promoting improved analytical tools and methods for analyzing national security planning and the allocation of resources.
- Ensure that the costs of DoD programs, including classified programs, are presented accurately and completely.
- Assess effects of DoD spending on the U.S. economy, and evaluate alternative policies to ensure that DoD programs can be implemented efficiently.

Step 1B: Develop Estimating Plan

After establishing the purpose of the software estimate, a plan for developing the estimate should be established.

The estimate development is itself a mini-project, and the associated plan should have all the components associated with sound project management.



Step 1B: Develop Estimating Plan, Cont.

In addition to the components listed below, the plan should also address, at least tentatively, cost estimating considerations such as data collection and planned estimating methodolgies. Select each component below to learn more.

Staffing | Requirements | Resources | Interfaces and Dependencies

Timeline

When is the estimate due, and are there any interim deliverables or reviews?

Popup Text

Timeline

When is the estimate due, and are there any interim deliverables or reviews?

Staffing

Who are the estimating team, and what are their availability and other commitments?

What is their level of expertise relative to software?

Requirements

What issues and scope (driven by the estimate purpose from the previous step) does the estimate need to address?

How detailed does the estimate need to be?

Resources

What estimating databases and tools are available?

Interfaces and Dependencies

What is the relationship between the software estimate and the total program estimate?

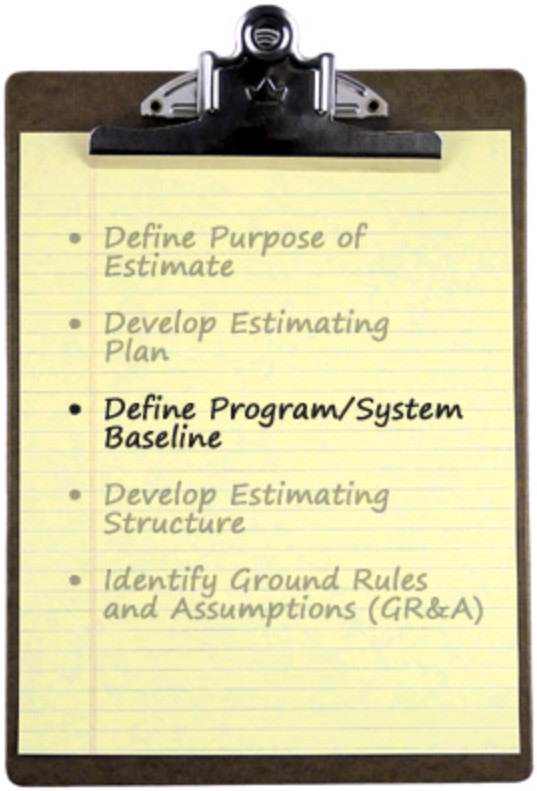
Step 1C: Define Program/System Baseline

A successful software estimate rests in large part on a thorough and accurate description of the system to be estimated and the program as part of which it is to be acquired. The standard mechanism for this is the Cost Analysis Requirements Description (CARD).

A CARD is required for MDAPs and MAIS programs, and CARD-like documentation is strongly encouraged for all programs. A single CARD will generally cover the entire system, including the software components, so it is important to be able to extract key pieces of information to support the software estimate.

The standard CARD outline is shown in Chapter 1 of the DoD 5000.4 Manual. The remainder of this step focuses on the software-related aspects. Section 1.2.2 Software Description recommends a DoD Form 2630 – now the Software Resources Data Report (SRDR) – to provide "information on the factors that will influence software development and maintenance costs."

Key Information



Popup Text

Cost Analysis Requirements Description (CARD)

A description of the technical and programmatic features of an acquisition program that is used as a common reference point by the teams preparing the life cycle cost estimates.

Key Information

The key point here is that program objectives and requirements must be clearly and precisely identified in order to develop good estimates of effort and cost early in the program's life cycle. The objectives of the program define what the end product is, what the end product's intermediate steps are, and when and for whom the system will be delivered. The objectives also help define how and by whom the project will be carried out.

Step 1C: Define Program/System Baseline - Software Intensive System Types

It is important to be clear about what type of software-intensive system (SIS) is being estimated. A SIS is defined as system in which software represents the largest segment in one or more of the following criteria: system development cost, system development risk, system functionality, or development time.

The type of system may affect the nature of the hardware and software and how they interact, the complexity of the software, the prevalence of COTS software, and so on. From an estimating perspective, typically the focus is on gathering data from programs of the same system type.

The three types of software-intensive systems are: Weapon Systems with Embedded Software, C4ISR Systems and Automated Information Systems (AIS). Each is discussed in more detail on the following pages.

Step 1C: Define Program/System Baseline - Software Intensive System Types - Embedded Software

Embedded software can be very tightly coupled with a weapon system such as a missile, or more loosely as in an avionics "box" on an aircraft.

These systems are more likely to involve custom-built hardware, and the associated software may have to run under memory or processor speed restrictions.

They may also involve firmware, wherein read-only software resides on a hardware device and cannot be readily modified under program control. In general, the software development may have more of an electrical engineering (EE) "flavor."

Popup Text

Firmware

The combination of a hardware device and computer instructions or computer data that reside as read-only software on the hardware device. The software cannot be readily modified under program control.

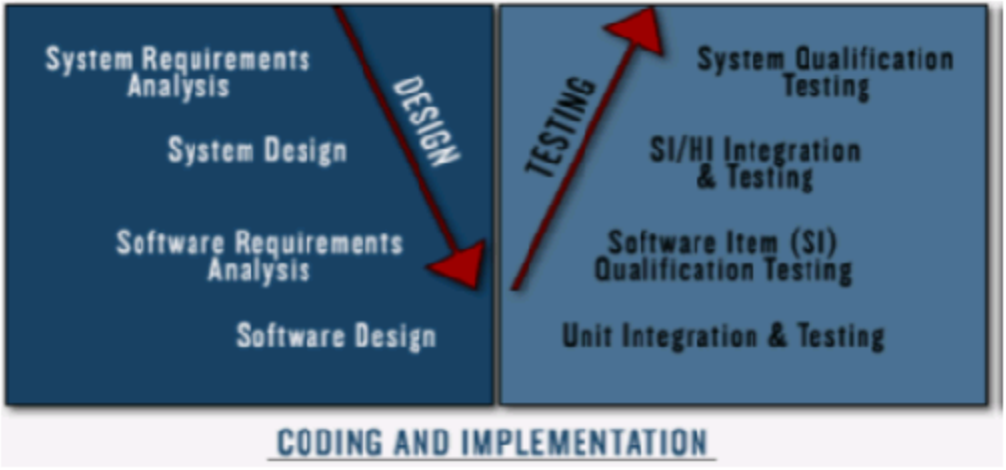
Step 1C: Define Program/System Baseline - Software Intensive System Types - Embedded Software, Cont.

In developing the software estimate for Embedded systems, it is important to understand the overall architecture and verify the allocation of system requirements to software. This is done via the Systems Engineering (SE) process, as depicted in the accompanying V-diagram.

Mission assurance and safety requirements may drive up the quality, reliability, and complexity of the software, and consequently the associated development effort.

It might be worth investigating whether there are statistically significant differences between software development for the particular type of weapon system at hand and other embedded software development.

There may be significant infrastructure costs (and even developed software!) for simulators and other testing and training devices.



Popup Text

Systems Engineering (SE)

The overarching process that a program team applies to transition from a stated capability to an operationally effective and suitable system. SE encompasses the application of SE processes across the acquisition life cycle (adapted to each and every phase) and is intended to be the integrating mechanism for balanced solutions addressing capability needs, design considerations and constraints, as well as limitations imposed by technology, budget, and schedule. The SE processes are applied early in concept definition, and then continuously throughout the total life cycle.

Step 1C: Define Program/System Baseline - Software Intensive System Types - C4ISR Systems

Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) systems are fielded military systems that support the warfighter by providing one or more of the indicated functions within the "battle space."

Software for these systems is often similar to software for weapons, with software for a communications satellite operating under comparable constraints, for example.

Popup Text

Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR)

Command and Control refers to the ability of military commanders to direct forces. The addition of Communications to the grouping reflects the fact that communications is required to enable this coordination. In modern warfare, Computers have become a key component as cyberspace is now seen as the "fifth dimension of warfare." Intelligence, Surveillance, and Reconnaissance involve methods of observing the enemy and one's area of operations.

Step 1C: Define Program/System Baseline - Software Intensive System Types - C4ISR Systems, Cont.

By contrast, commodity hardware is much more common, though it often needs to be ruggedized, as in ensuring that a laptop can operate in the extreme heat of an inhospitable desert environment or that the servers in racks adjacent to a ship's combat information center (CIC) can withstand anticipated pitch, roll, and yaw.

C4ISR Systems often require frequent software updates and lots of real-time "hooks" into other systems.

In our example ProRad program, the software-defined radios, whose primary function is Communications, are considered C4ISR Systems.

Note that while "front-end" ISR systems such as satellites and airborne sensors are typical of this kind of system, "back-end" intelligence systems performing tasking, processing, exploitation, and dissemination (TPED) may have more in common from a technical perspective with AIS, discussed next.

Popup Text

Tasking, Processing, Exploitation, and Dissemination (TPED)

The process of assigning intelligence, surveillance, and reconnaissance (ISR) assets to targets, utilizing the resulting information, and distributing it to those with a need to know in order to produce a strategic or tactical advantage.

Step 1C: Define Program/System Baseline - Software Intensive System Types - Automated Information Systems (AIS)

An automated information system (AIS) is an acquisition program that acquires information technology that is not embedded in a weapon system.

AIS programs normally are involved with and directly related to information storage, processing, and display — requiring resources for hardware, software, data, telecommunications, etc. AIS programs that meet the specified dollar thresholds in DoD Instruction 5000.2, Enclosure 3, qualify as MAIS.

Popup Text

Automated Information System (AIS)

A combination of computer hardware and computer software, data, and/or telecommunications that performs functions such as collecting, processing, storing, transmitting, and displaying information. Excluded are computer resources, both hardware and software, that are an integral part of a weapons system; used for highly sensitive classified program as determined by the Secretary of Defense; used for other highly sensitive information technology (IT) programs as determined by the Under Secretary of Defense for Acquisition, Technology and Logistics) (USD(AT&L)) or designee to be better overseen as a non-AIS program (e.g., a program with a low ratio of research, development, test, and evaluation (RDT&E) funding to total program acquisition costs or that requires significant hardware development).

Step 1C: Define Program/System Baseline - Software Intensive System Types - Automated Information Systems (AIS), Cont.

AIS are generally business systems that support the operation of the department and are housed in typical office buildings or data centers.

These systems handle personnel information, financial transactions, and the like. Increasingly, these systems are being implemented using commercially-available Enterprise Resource Planning (ERP) software packages, which transfers the bulk of the cost from software development to software configuration and business process re-engineering (BPR).

AIS generally run on commodity hardware and may take advantage of commercial technology such as server virtualization.

Even when they do not involve ERP systems, they tend to be more COTS-intensive, though DoD's security and continuity of operations (COOP) requirements may drive up costs compared to commercial analogues.

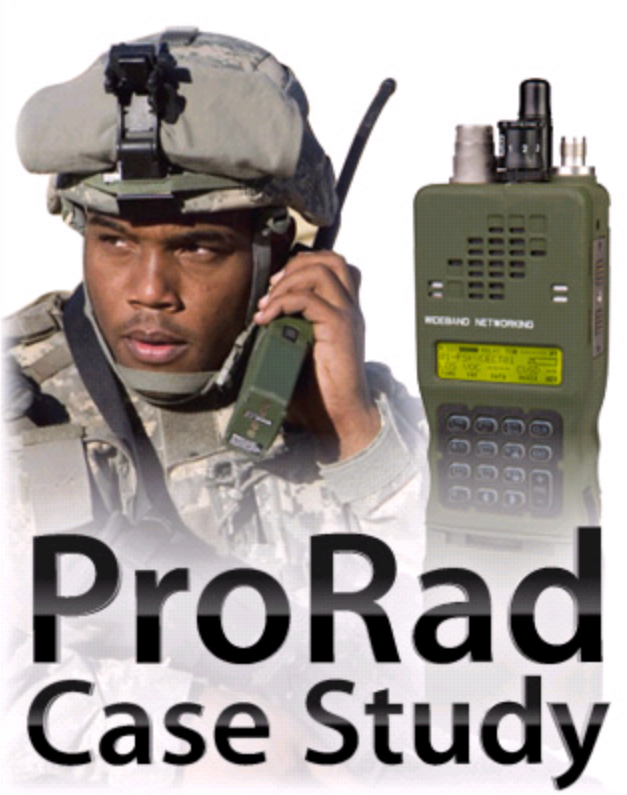
Step 1C: Define Program/System Baseline - ProRad Program/System Definition

In 1991, the term software-defined radio (SDR) was coined to describe radio devices implemented in software and running on generic hardware. This idea of software-programmable radio technology was a departure from traditional, hardware-specific radio architectures, and offered considerable improvements in flexibility, interoperability, and upgradeability.

Before these multi-band, multi-mode radios could be fully implemented, however, advancements were necessary in the processing, analog/digital (A/D), and radio frequency (RF) hardware. These radios were not targeted for development until the year 2000 or beyond.

Aside from the hardware challenges of SDR, a major part of the ProRad effort was to develop or acquire the software for about 31 different waveforms that were to be hosted on the ProRad.

To do this, the Joint ProRad program office had to first develop a software standard called the software communications architecture (SCA), intended to standardize the software's operating environment, and the control and communication mechanisms for both the hardware and the external interfaces of the radio.



Step 1C: Define Program/System Baseline - Predecessor Systems

CARD Section 1.5 Predecessor and/or Reference System describes the currently operational or pre-existing system with a mission similar to that of the proposed system. It is often the system being replaced or augmented by the new acquisition.

The discussion should identify key system-level characteristics of both the predecessor and/or reference system and the new or proposed system.

Any problems associated with the predecessor system should be discussed, along with any significant differences between the predecessor system and the proposed system.

The narrative should also describe how the predecessor system is to be replaced with the proposed system (e.g., one-for-one replacements, etc.). Information on the planned disposition of the replaced systems should be provided so that disposal costs and benefits can be considered in the cost estimate.

The above information should also be provided on analogous subsystems and components that can be used to scope or estimate the new system.

Step 1C: Define Program/System Baseline - Predecessor Systems, Cont.

The predecessor system may also be referred to as the legacy or historical system. The comparison focuses on the capability gap that will be addressed by the new system.

The predecessor system is a prime candidate for both analogies for the three key components of the software estimate (sizing, cost, and schedule) and subject matter experts (SMEs) that can help with the interpretation of historical data and technical aspects of the program that may impact cost or schedule.

As previously noted, the rapid technological change associated with software and AIS can make drawing these analogies a challenge, even to predecessor programs only a few years old, but don't buy the "blank sheet of paper" theory. There is always value in examining and using data from the predecessor system.

D

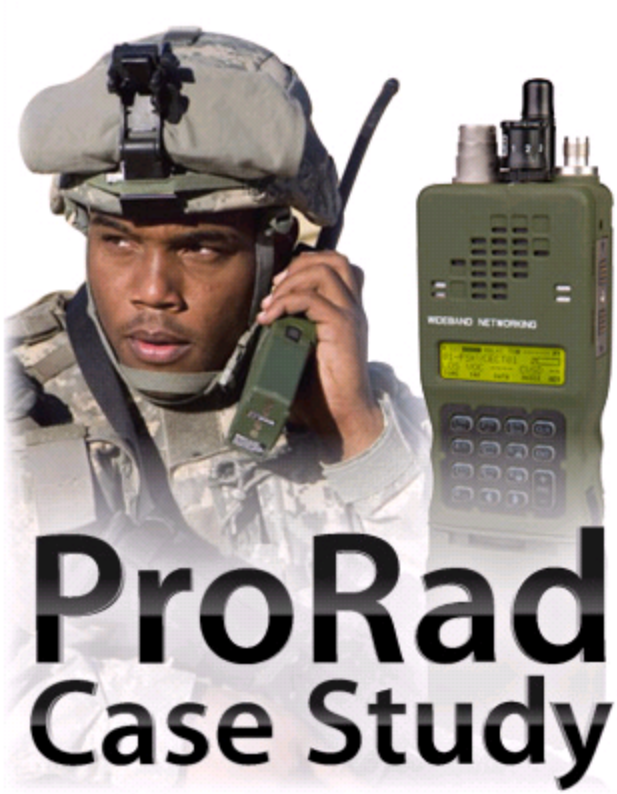
Long Description

Continuum line labeled Legacy on left end New on right end. Underneath the line is a bracket labeled Capability Gap.

Step 1C: Define Program/System Baseline - Predecessor Systems - ProRad Predecessor System

The Joint ProRad Program Office drew primary source data for its waveform development cost estimates from these programs:

- Army Systems: Joint Tactical Terminal (JTT); Joint Communications Interface Terminal (JCIT)
- Navy Systems: Digital Modular Radio (DMR); Multi-Function Information Distribution System (MIDS)
- Air Force Systems: Airborne Integrated Terminal Group (AITG); SpeakEASY Radio



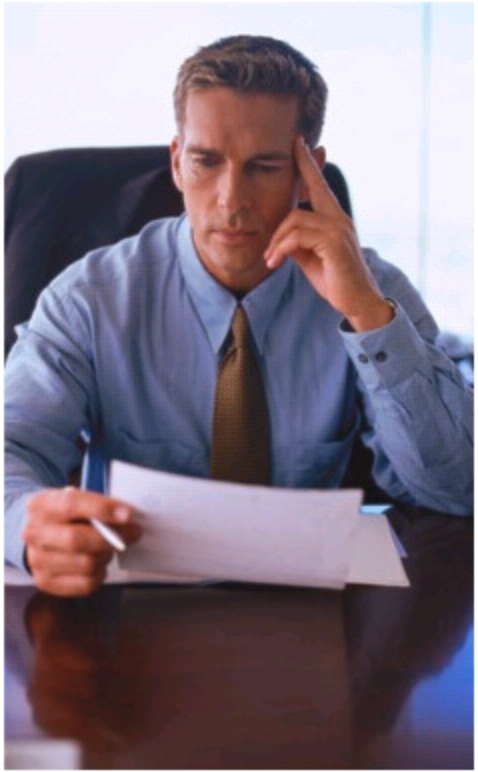
Step 1C: Define Program/System Baseline - Acquisition Strategy

CARD Section 8.0 Acquisition Plan and/or Strategy describes the acquisition plan for the system, addressing both Contractors and Contract Type.

It should identify the number of prime contractors expected to compete during each acquisition phase. The specific contractors and subcontractors involved in each phase should be identified, if known.

It should also describe the type of contracts to be awarded in each phase of the program and discuss the status of any existing contracts.

Contract types vary according to the degree and timing of responsibility (i.e., risk) assumed by the contractor for the cost of performing the contracted work effort; and the amount and nature of the profit incentive offered the contractor to achieve or exceed specified standards or goals set by the government.



Step 1C: Define Program/System Baseline - Acquisition Strategy, Cont.

The analyst should make sure that the acquisition strategy brings to bear the appropriate development and integration skills needed to supplement any functionality provided by COTS software and other non-developmental items (NDIs).

It is important to understand who is doing the work, whether a single contractor, multiple contractors, or even a contractor/government team. Specialized skills required for software and AIS often require multiple subcontractors.

Where specific organizations are known, software estimates should reflect their specific productivities and rates. Where they are not known, as early in the program, sufficient uncertainty should be incorporated to capture the range of possible values across the industrial base.

Contract types are grouped into two broad categories: Fixed-price and Cost-reimbursement.

The contract type should be chosen to reflect the nature of work being performed and associated risk. Fixed-price contracts are not generally appropriate for software development.

Popup Text

Non-Developmental Item (NDI)

1. An NDI is any previously developed item of supply used exclusively for government purposes by a federal agency, a State or local government, or a foreign government with which the United States has a mutual defense cooperation agreement.
2. Any item described in item 1 that requires only minor modifications or modifications of the type customarily available in the commercial marketplace in order to meet the requirements of the procuring department or agency.
3. Any item of supply being produced that does not meet the requirements of items 1 or 2 solely because the item is not yet in use.

Fixed-price

A type of contract that provides for a firm price to the government, or in appropriate cases, an adjustable price.

Cost-reimbursement

A type of contract that provides for payment to the contractor of allowable costs incurred in the performance of the contract, to the extent prescribed in the contract. This type of contract establishes an estimate of total cost for the purpose of obligating of funds and establishes a ceiling that the contractor may not exceed without prior approval of the contracting officer (CO).

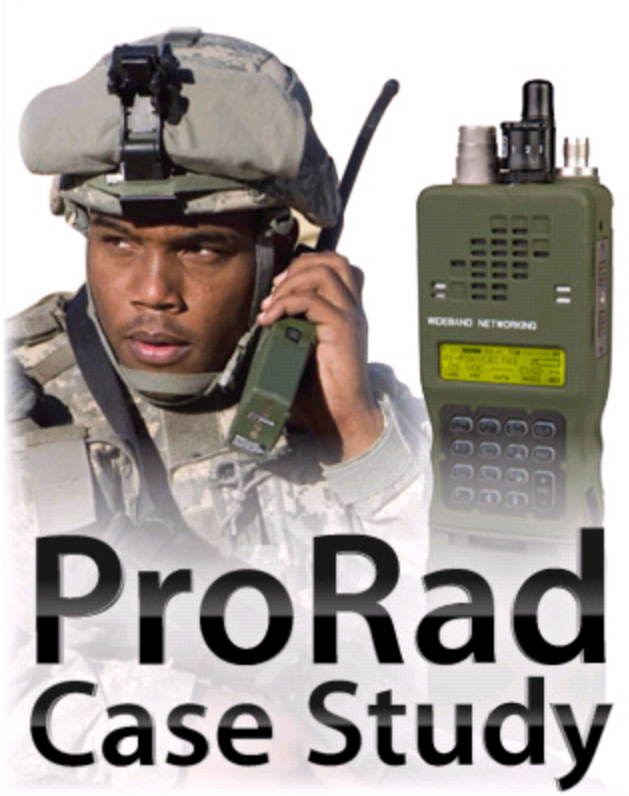
Step 1C: Define Program/System Baseline - Acquisition Strategy - ProRad Acquisition Strategy

Cost Plus Award Fee (CPAF) contracts were planned for development of the Software Communications Architecture (SCA). This was leading-edge work as design standards for software-defined radios had not yet been established by the commercial sector.

Firm Fixed Price (FFP) contracts were planned for waveform development since the waveforms are known and the SCA would be provided to the vendor. Waveform development included all design, coding, and factory testing by the vendor, and support to all Joint ProRad Program Office validation and certification efforts.

In addition, the Program Office planned to purchase all necessary test and waveform validation tools, waveform application software, and cryptographic algorithms for programmable cryptographic devices.

Note that even though the waveform development was considered to be low enough risk for fixed-priced contracting once the SCA is provided, there may be considerable risk and uncertainty early in the program, including downstream impacts on waveform development, before SCA development is complete.



Popup Text

Contract, Cost Plus Award Fee (CPAF)

A cost reimbursement type contract suitable for level of effort contracts where mission feasibility is established but measurement of achievement must be by subjective evaluation rather than objective measurement.

A CPAF contract may not be used to avoid establishing a Cost Plus Fixed Fee (CPFF) contract when the criteria for CPFF contracts apply or developing objective targets so a Cost Plus Incentive Fee (CPIF) contract can be used.

Contract, Firm, Fixed Price (FFP)

Provides for a price that is not subject to any adjustment on the basis of the contractor's cost experience in performing the contract. This type of contract places upon the contractor maximum risk and full responsibility for all costs and resulting profit or loss. Provides maximum incentive for the contractor to control costs and imposes a minimum administrative burden on the government.

Step 1C: Define Program/System Baseline - System Architecture

Within one of three previously described software-intensive system types, architecture defines functionality of subsystems and internal and external interfaces, and a clear understanding of this architecture is essential for an accurate software estimate.

The Department of Defense Architecture Framework (DoDAF), with its complement of operational views (OV), systems views (SV), technical standards views (TV), and all views (AV), is the preferred means of expressing the system architecture.

CARD Section 1.1.2. System Functional Relationships describes the "top-level" functional and physical relationships among the subsystems within the system as well as the system's relationship to other systems. Section 1.1.3 System Configuration identifies the equipment (hardware and software) work breakdown structure (WBS) for the system.

The key is to have a solid basis for the requirements that have been allocated to the system's software, as opposed to requirements that are addressed by other components of the system (i.e., hardware) or by external systems.

Interface definitions for those external systems are also important, as the number and complexity of interfaces required may drive both software development and testing. The nature of user interfaces should be considered, as human-computer interface (HCI) is often a design and cost driver as well.

The following pages address functionality provided by developed software versus COTS software, including ERP.

Department of Defense Architecture Framework (DoDAF)

The DoDAF provides the guidance and rules for developing, representing, and understanding architectures based on a common denominator across DoD, Joint, and multinational boundaries. It provides insight for external stakeholders into how the DoD develops architectures. The DoDAF is intended to ensure that architecture descriptions can be compared and related across programs, mission areas, and, ultimately, the enterprise, thus, establishing the foundation for analyses that supports decision-making processes throughout the DoD.

Human-Computer Interface (HCI)/Man-Machine Interface (MMI)

Degree of compatibility between the user (individual) and the equipment being used.

Questions Managers Should Ask

Have steps been taken to ensure the integrity of the estimating process?

- Have memoranda of agreement (MOAs) been completed and signed with the organizations whose contributions affect cost or schedule?

Step 1C: Define Program/System Baseline - System Architecture - Hardware vs. Software

First take a moment to note similarities and differences between hardware and software from an estimator's perspective. They both follow a "design-develop-test" process and have support requirements (maintenance and upgrades) once fielded.

However, whereas hardware is built multiple times, there is no Production phase to speak of in software; you can simply generate a copy of the developed code.

You will never see a T1 or a learning curve in software. They have similar general drivers in terms of scope (size of the object being produced), complexity (sophistication of the object being produced), and capability (skills and tools being brought to the task).

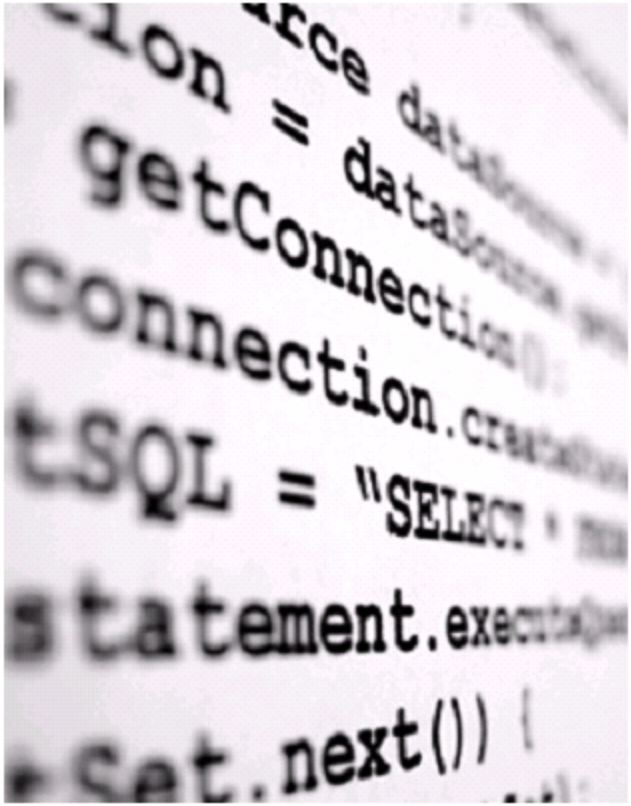
In the Joint ProRad software-defined radio example, the SCA and waveform processing are two crucial pieces of software functionality that are required for the radios to work with a number of different hardware configurations.

Step 1C: Define Program/System Baseline - System Architecture - Developed Software

The traditional approach to software is to develop so-called "custom software," often taking advantage of the ability to reuse code from previous similar efforts or, with the advent of object-oriented programming, from reuse libraries. Most databases and models are geared toward this kind of software development.

Developed software is broken down into Software Items (SIs) (also known as Computer Software Configuration Items (CSCIs)), which are further decomposed into Computer Software Components (CSCs) and Computer Software Units (CSUs). (The "Unit" in the development process step Code and Unit Test refers to the latter.) CSUs are typically separately compilable pieces of code, and may also be called modules.

The system architecture should make it clear which components are envisaged to be developed software.



Popup Text

Software Item (SI)

An aggregation of software, such as a computer program or database, that satisfies an end-use function and is designated for purposes of specification, qualification, testing, interfacing, configuration management, or other purposes. An SI is made up of Computer Software Units (CSUs).

Computer Software Configuration Item (CSCI)

Under some software development standards, an aggregation of software that is designated for configuration management (CM) and treated as a single entity in the CM process.

Computer Software Component (CSC)

Under some software development standards, a functional or logically distinct part of a Computer Software Configuration Item (CSCI) or Software Configuration Item (SCI). A CSC is typically an aggregate of two or more Computer Software Units (CSUs).

Computer Software Unit (CSU)

Under some software standards, the smallest subdivision of a Computer Software Configuration Item (CSCI) for the purposes of engineering management. CSUs are typically separately compliable pieces of code.

Module

An independently compilable software component made up of one or more procedures or routines or a combination of procedures and routines.

Step 1C: Define Program/System Baseline - System Architecture - COTS Software

In many cases, commercial off-the-shelf (COTS) software can provide "out-of-the-box functionality" to address key software requirements. COTS is integrated via the use of Application Programmer Interfaces, or APIs, and so-called "glue code." If using COTS software, be aware of these issues:

- COTS packages often use **proprietary development languages**.
- It may be **difficult to customize** the COTS package to address additional requirements.
- The COTS package might not have been designed to accept **changing requirements**.
- It might be necessary to perform **additional testing** to ensure the COTS package will work as expected.
- **Upgrades may require reintegration** rather than just being loaded.
- **Support costs** can be affected by COTS use. (The "C" in COTS does not stand for "Cheap"!)
- You **must still perform the functions of requirements definition, design, and test** when using COTS packages.
- **Licensing, royalties, incompatibilities with other packages, and lack of access to the source code** are among many issues unique to COTS software implementations.
- **Estimating COTS integration** has its own difficulties beyond the engineering and management challenges we've already mentioned.

"Quotations from Chairman David: A Little Red Book of Truths to Enlighten and Guide on the Long March Toward the COTS Revolution," David J. Carney, CMU/SEI, 1998.

Popup Text

Commercially Available Off-The-Shelf (COTS)

A commercial item (CI) sold in substantial quantities in the commercial marketplace and offered to the government under a contract or subcontract at any tier, without modification, in the same form in which it was sold in the marketplace. This definition does not include bulk cargo such as agricultural products or petroleum.

Step 1C: Define Program/System Baseline - System Architecture - ERP Software

A particular kind of COTS software, Enterprise Resource Planning (ERP), has become prevalent in AIS implementation.

An ERP system is a business support system that maintains in a single database the data needed for a variety of business functions and is based on a modular software design. SAP and Oracle are two examples of major ERP software providers.

ERP estimates must address costs for Gap Analysis, Business Process Re-engineering (BPR), COTS Integration, and Model Development, in addition to the traditional estimation of the custom code development. A common driver for ERP cost is the number of Reports, Interfaces, Conversions, Extensions, Forms, and Workflows (RICE-FW).

Database Development, whether part of an ERP implementation or not, is an important software estimating consideration not included in the typical development process.

For example, a satellite may have data tables for the requisite star tracker data, which are stored in the same memory as the developed code but would not be estimated using the same productivity.

Popup Text

Enterprise Resource Planning (ERP)

A single business support system that provides for a variety of business functions.

Business Process Re-engineering (BPR)

An approach aiming at improvements by means of elevating efficiency and effectiveness of the business process within and across organizations. The key to BPR is for organizations to look at their business processes from a "clean slate" perspective and determine how they can best construct these processes to improve how they conduct business. Also known as BPR, Business Process Redesign, Business Transformation, or Business Process Change Management.

Reports, Interfaces, Conversions, Extensions, Forms, and Workflows (RICE-FW)

Count of Tailored Objects commonly used as a cost driver for Enterprise Resource Planning (ERP) system implementation. Some earlier sources refer to RICE objects, omitting Forms and Workflows.

Step 1C: Define Program/System Baseline - Development Approach

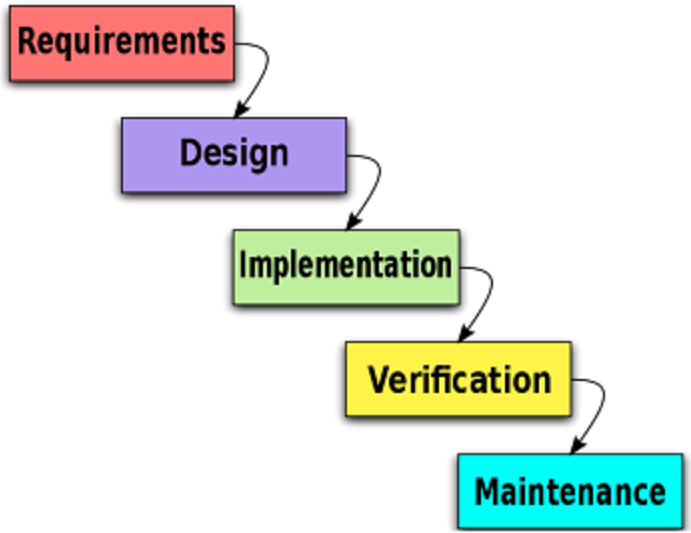
CARD Section 1.2.2 Software Description describes the software resources associated with the system. It should distinguish among operational, application, and support software and identify which items must be developed and which can be acquired off-the-shelf.

For each software sub-element, sections address Host Computer Hardware Description, Programming Description, Design and Coding Constraints, and Commonality.

An important consideration is how the software will be developed. The development approach may include use of object-oriented (OO) programming techniques, use of Computer-Aided Software Engineering (CASE) tools, degree of reliance on COTS, and development paradigm used.

The four main Software Development Paradigms are: Waterfall, Incremental, Evolutionary, and Spiral.

The paradigm chosen is often related the Acquisition Strategy. Some software development efforts may use even newer techniques such as Agile (aka Scrum).



Popup Text

Computer- Aided Software Engineering (CASE)

The use of computers to aid in the software engineering process. CASE tools may include the application of software tools to software design, requirements tracing, code production, testing, document generation, and other software engineering activities. Assemblers and compilers are CASE tools.

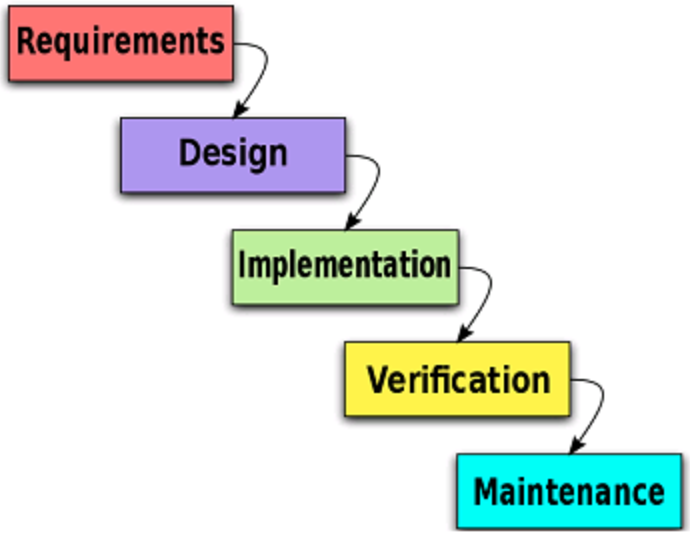
Step 1C: Define Program/System Baseline - Development Approach - Software Development Paradigms

The four main Software Development Paradigms are Waterfall, Incremental, Evoluntary, and Spiral. Select each tab below to learn more about each.

Incremental | Evolutionary | Spiral

The Waterfall Paradigm, also called "Grand Design," accomplishes the development in a single pass of the typical systems engineering V-process, including Requirements; Design; Code and Unit Test; CSC and CSCI Integration and Test; and System Test.

It is most appropriate when requirements are stable and no interim capable is needed more quickly.



The importance of Life Cycle Methodology for the software estimate is that it directly affects how the estimate is structured and how the process activities are mapped to effort. For example, "fixed" one-time costs such as up-front requirements and final test must be separated out from "variable" costs for activities that repeat with each iteration.

Popup Text

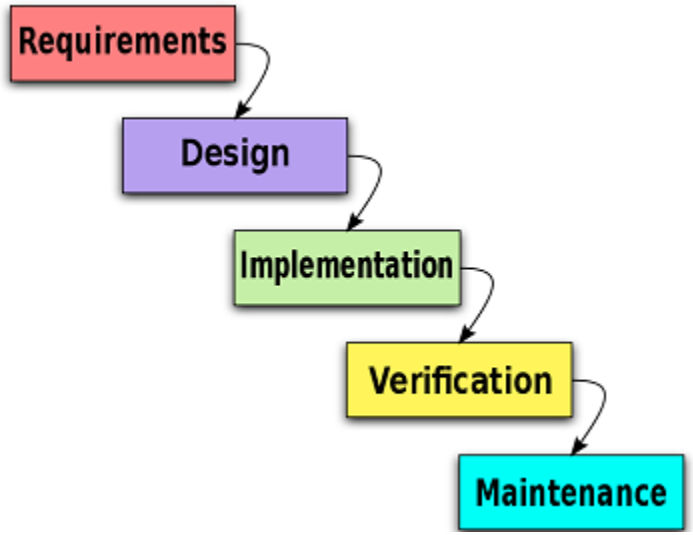
Waterfall

The Waterfall Paradigm, also called “Grand Design,” accomplishes the development in a single pass of the typical systems engineering V-process, including Requirements; Design; Code and Unit Test; CSC and CSCI Integration and Test; and System Test.

It is most appropriate when requirements are stable and no interim capable is needed more quickly.

Waterfall

Development activities are performed in order, with possibly minor overlap, but with little or no iteration between activities. User needs are determined, requirements are defined, and the full system is designed, built, and tested for ultimate delivery at one point in time. A document-driven approach best suited for highly precedential systems with stable requirements.



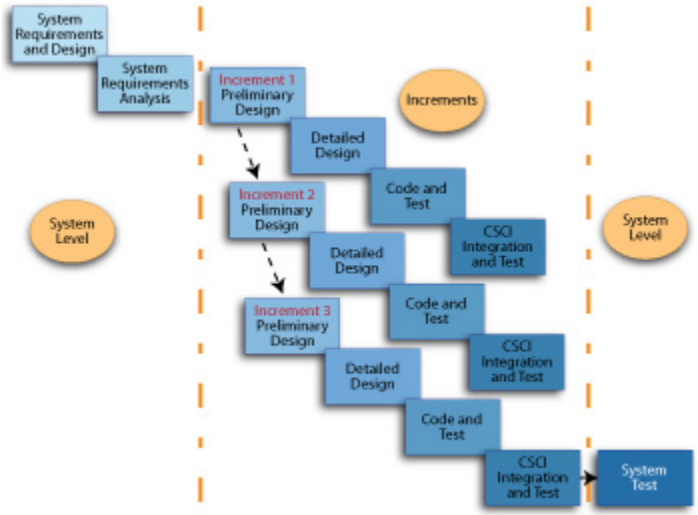
Incremental

Both Incremental and Evolutionary divide the development up into individual increments or releases to provide some interim capability.

In Incremental, the requirements are stable up front and allocated to the increments, which may be sequential but are often overlapping.

Incremental

Determines user needs and defines the overall architecture, but then delivers the system in a series of increments ("software builds"). The first build incorporates a part of the total planned capabilities; the next build adds more capabilities, and so on, until the entire system is complete.



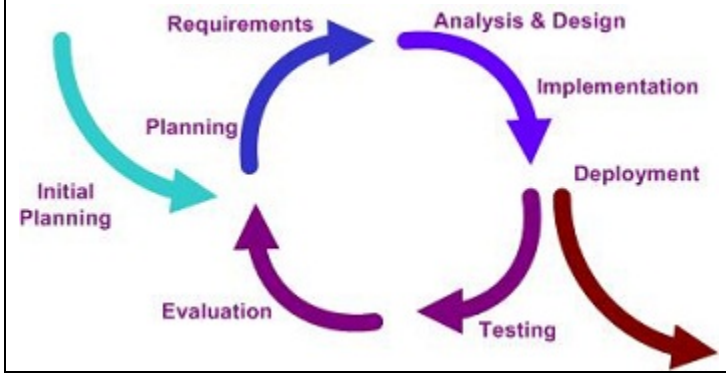
Popup Text

Evolutionary

Evolutionary, in contrast to incremental, has only an initial set of requirements for the first build, and the requirements evolve with each in a series of non-overlapping releases, the so-called “build a little, test a little” approach.

Evolutionary

A development strategy which begins with a prototype containing core capability. Customer provides feedback, prototype is adjusted, and additional capability added. Process is repeated until the system is complete.

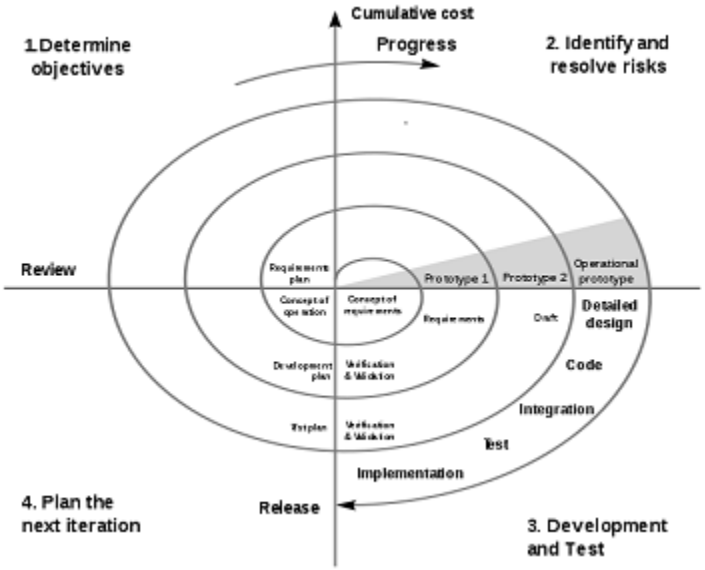


Spiral

Spiral, is similar to Evolutionary, with several iterations, each building on the previous, but with added steps for risk assessment.

Spiral

A risk-driven controlled prototyping approach that develops prototypes early in the development process to specifically address risk areas followed by assessment of prototyping results and further determination of risk areas to prototype. Areas that are prototyped frequently include user requirements and algorithm performance. Prototyping continues until high risk areas are resolved and mitigated to an acceptable level.



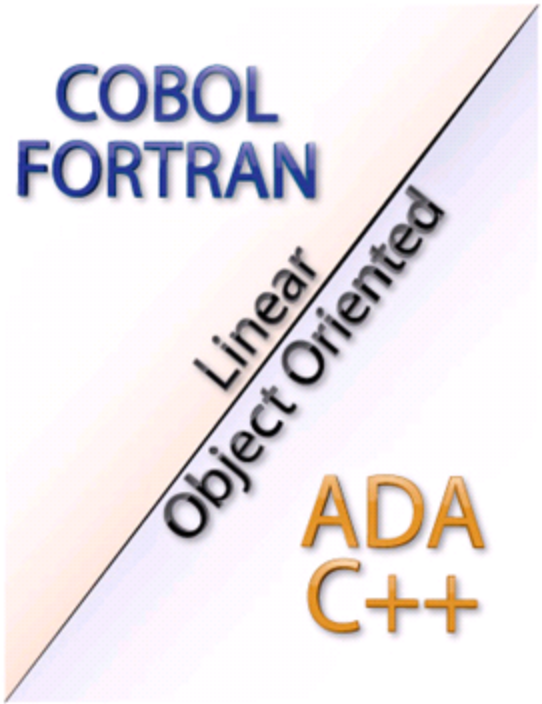
Step 1C: Define Program/System Baseline - Development Approach - Programming Paradigms

The traditional Linear approach to computer programming, wherein code is "custom built" line by line, has largely been superseded by Object-Oriented (OO) programming, which makes use of pre-built, standardized, interchangeable objects and libraries of functions that operate on them.

Traditional Linear Programming languages include COBOL and FORTRAN, whereas Ada and C++ are typical of the OO paradigm.

The importance of Programming Paradigms for the software estimate lies in which analogous programs to pick.

Programs developed under one paradigm would likely not be representative of the effort needed for development under the other paradigm.



D

Popup Text

Object-Oriented (OO)

System made up of pre-built, standardized, interchangeable objects.

Long Description

A split graphic with COBOL and FORTRAN in half titled Linear and ADA and C++ in half titled Object Oriented.

Step 1C: Define Program/System Baseline - Testing Approach

CARD Section 9.0 System Development Plan addresses software reuse from phase to phase and testing, both Developmental Test and Evaluation (DT&E) in Section 9.2 and Operational Test and Evaluation (OT&E) in Section 9.3.

The number, type, location, and expected duration of tests (for both hardware and software) should be identified, along with the organizations that will conduct the test programs.

It is important to distinguish between Software Testing, which addresses only software functionality, and subsequent System Testing, which addresses functionality of the total system and would entail diagnosis of whether any detected failures were due to hardware, software, or a combination thereof. Both are iterative in nature.

Software Testing may include peer reviews, unit test (of individual CSUs), and qualification test (of CSCIs).

System Testing may include functional configuration audits (FCAs) and physical configuration audits (PCAs).

Popup Text

Developmental Test and Evaluation (DT&E)

1. Any testing used to assist in the development and maturation of products, product elements, or manufacturing or support processes.
2. Any engineering-type test used to verify status of technical progress, verify that design risks are minimized, substantiate achievement of contract technical performance, and certify readiness for initial operational testing. Developmental tests generally require instrumentation and measurements and are accomplished by engineers, technicians, or soldier operator-maintainer test personnel in a controlled environment to facilitate failure analysis.

Operational Test and Evaluation (OT&E)

The field test, under realistic conditions, of any item (or key component) of weapons, equipment, or munitions for the purpose of determining the effectiveness and suitability of the weapons, equipment, or munitions for use in combat by typical military users; and the evaluation of the results of such tests.

Functional Configuration Audit (FCA)

Verifies that all item or subsystem requirements established in the functional and allocated baselines, specifications, and test plans have been tested successfully, and corrective action has been initiated, as necessary.

Physical Configuration Audit (PCA)

Physical examination of the actual configuration of the item being produced. It verifies that the related design documentation matches the item as specified in the contract.

Step 1C: Define Program/System Baseline - Support Approach

CARD Section 3.4.1.2 Software Support Concept describes the software support concept, including methods planned for upgrades and technology insertions. The discussion should also address post-development software support (PDSS) requirements.

PDSS encompasses the support needed for software during the operations and support (O&S) phase and the associated costs. PDSS includes but is not limited to software maintenance, which is discussed in more detail on the next page. Additional effort is entailed in gathering, diagnosing, and reporting problems and issues, which may need to be addressed by software maintenance, hardware maintenance, or a combination thereof.

The help desk function that takes calls and files trouble tickets is an example of PDSS effort not included in software maintenance. There are also related Sustainment Engineering efforts. Software cost estimating models generally only account for software maintenance costs, and these additional areas need to be addressed outside of the model.

The software support approach should at a minimum indicate whether an organic approach is planned, with support provide by the appropriate government depot, or a contractor logistics support (CLS) approach, where the maintainer is usually the original system developer.

It is also important to specify whether a performance-based logistics (PBL) approach will be taken, and if so, which performance criteria are being recommended. PBL may be implemented with both organic and contractor options.

Popup Text

Post-Deployment Software Support (PDSS)

Those software support activities that occur after the deployment of the system.

Operations and Support (O&S) Phase

The fifth phase of the life cycle as defined and established by DoDI 5000.02 after Materiel Solution Analysis (MSA), Technology Development (TD), Engineering and Manufacturing Development (EMD), and Production and Deployment (P&D). This phase consists of two efforts, Life Cycle Sustainment and Disposal. The phase is not initiated by a formal milestone, but instead begins with the deployment of the first system to the field; an act that initiates the Life Cycle Sustainment effort of this phase. The Life Cycle Sustainment effort overlaps the Full-Rate Production and Deployment (FRP&D) effort of the P&D phase.

Sustainment Engineering

Technical effort required to support an in-service system in its operational environment to ensure continued operation and maintenance of the system with managed risk, including:

- Collection and evaluation of service use and maintenance data and root cause analysis of in-service problems such as operational hazards, deficiency reports, parts obsolescence, corrosion effects, reliability and maintainability (R&M) trends, safety hazards, failure causes and effects, and operational usage profiles changes;
- Development of required design changes to resolve operational issues, introduction of new materials, and revising product, process, and test specifications;
- Oversight of the design configuration baselines to ensure continued certification compliance, and technical surveillance of critical safety items and approved sources for those items; and
- Periodic review of system performance against baseline requirements, analysis of trends, and development of management options and resource requirements for resolution.

Popup Text

Contractor Logistics Support (CLS)

The performance of maintenance and/or materiel management functions for a DoD system by a commercial activity. Current policy allows for the provision of system support by contractors on a long-term basis. Performance-Based Logistics (PBL) contracts should be used when utilizing CLS. Also called Long-Term Contractor Logistics Support.

Performance-Based Life Cycle Product Support / Performance-Based Logistics (PBL)
An outcome-based product support strategy that plans and delivers an integrated, affordable performance solution designed to optimally balance readiness and life-cycle costs by leveraging public and private industrial base capabilities. DoDI 5000.02 introduced the term “Product-Based Life Cycle Product Support” as the latest evolution of Performance-Based Logistics and stated that both terms can be referred to as “PBL.”

Step 1C: Define Program/System Baseline - Support Approach - Software Maintenance

Maintenance of software is different from maintenance of hardware in that software does not degrade or wear out the way that hardware does. Even though software is not typically considered to "break," continued use may uncover bugs that slipped through the testing phase. Software may also encounter new problems when introduced to an environment different from the one for which it was designed. New user requirements may also emerge. Software maintenance is notionally divided into the corrective, adaptive, perfective and preventive. Select the tabs below to learn more about each.

Adaptive | Perfective | Preventive

Corrective maintenance fixes bugs in software. Bugs are errors in the code that must be changed to make the software do what it is supposed to.

Popup Text

Corrective

Corrective maintenance fixes bugs in software. Bugs are errors in the code that must be changed to make the software do what it is supposed to.

Adaptive

Adaptive maintenance covers effort to accommodate new requirements such as changes to hardware or external interfaces, or upgrades to COTS software, any of which may force changes to the developed software.

Perfective

Perfective maintenance makes improvements to the software such as increasing its speed or reliability, or adds functionality.

These efforts are often “mini-developments,” with an up-front assessment of how many new requirements to “bite off” for a given planned maintenance release.

Preventive

Preventive maintenance entails performing activities to increase software maintainability and prevent problems in the future.

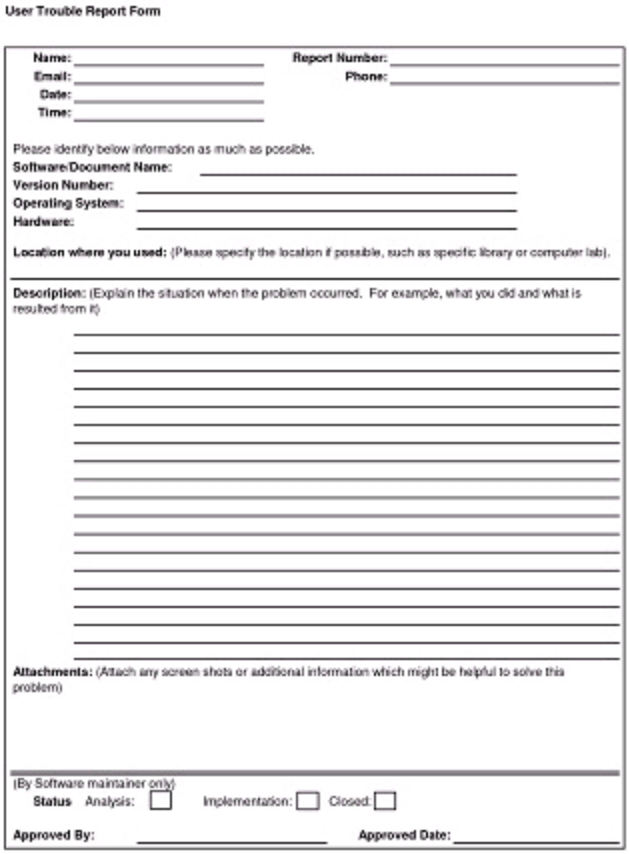
Step 1C: Define Program/System Baseline - Support Approach - Software Maintenance Reporting

Maintenance items that must be addressed are often catalogued as Software Trouble Reports (STRs) or some similar designation, which are usually categorized by priority or severity on a 1 to 5 scale, with 1 being critical and 5 a minor annoyance.

The simplest and most common approach to estimating software maintenance is to establish the level of effort (LOE) needed to maintain a given code base as a historical analogy scaled by size, which is a mathematically equivalent to dividing the estimated size of the delivered code by a SLOC/FTE rate.

Though the SDR approach is supposed to be hardware-independent, the ProRad program may find itself facing the need for adaptive maintenance to address changes in hardware or the SCA itself.

Select image to view an enlarged version



User Trouble Report Form

Name: _____ Report Number: _____
Email: _____ Phone: _____
Date: _____
Time: _____

Please identify below information as much as possible.
Software/Document Name: _____
Version Number: _____
Operating System: _____
Hardware: _____

Location where you used: (Please specify the location if possible, such as specific library or computer lab).

Description: (Explain the situation when the problem occurred. For example, what you did and what is resulted from it)

Attachments: (Attach any screen shots or additional information which might be helpful to solve this problem)

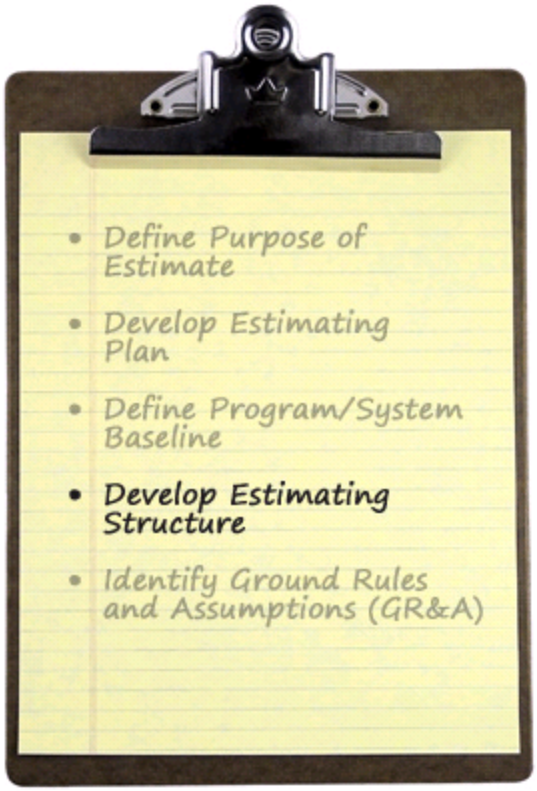
(By Software maintainer only)
Status Analysis: ☐ Implementation: ☐ Closed: ☐

Approved By: _____ **Approved Date:** _____

Step 1D: Develop Estimating Structure

At this point in the Develop Scope and Approach lesson, the purpose and the plan for the estimate have been established, and the program for the software-intensive system has been defined.

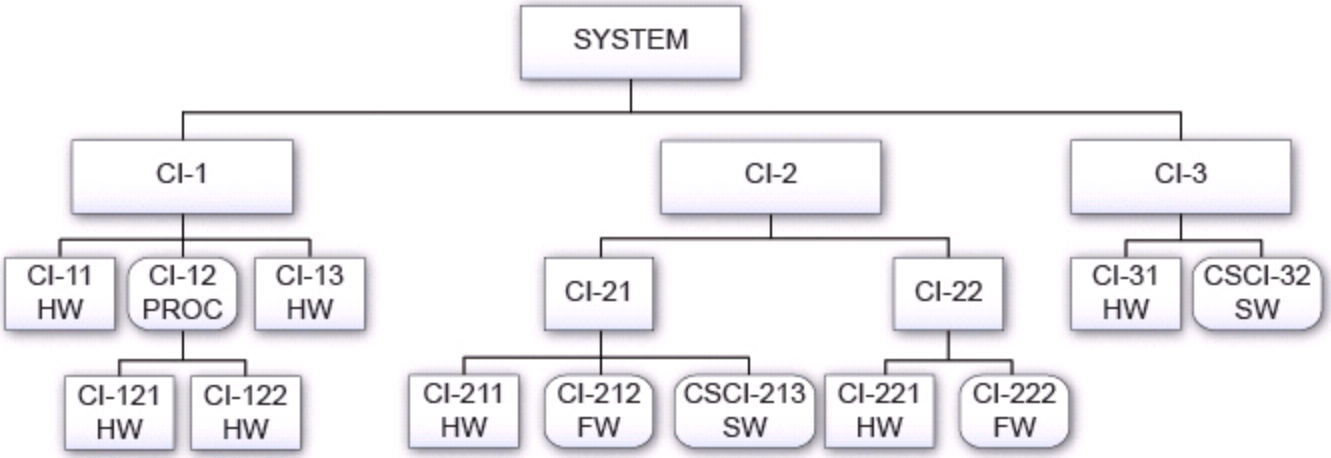
The next step in the process is to establish the structure for the cost estimate.



Step 1D: Develop Estimating Structure, Cont.

The Cost Element Structure (CES) for an LCCE generally consists of a Product-Oriented Work Breakdown Structure (WBS) for the Acquisition portion (Development plus Production, though the latter is negligible in the case of software) and a CES for O&S costs.

The goal of the LCCE CES is to be comprehensive, with neither omission of cost nor double-counting ("no gaps or overlaps"). For software estimates, there is a danger of focusing on Code & Unit Test to the exclusion of the myriad of supporting activities needed for a successful software development effort. It is important to remember that software is process-intensive, and therefore there is a need to account for all these processes in the estimate. An AIS CES may differ from a weapon system CES in that it may include phase-out costs for predecessor systems.



Popup Text

Cost Element Structure (CES)

A set of i) mutually exclusive and ii) exhaustive categories of cost that serve as the structure of (basis for) a cost estimate. Similar to a contract work break down structure and a work breakdown structure. Many CESs are standardized for use within DoD (and also in industry).

Work Breakdown Structure (WBS)

An organized method to break down a project into logical subdivisions or subprojects at lower and lower levels of details. It is very useful in organizing a project. See *Military Standard (MIL-STD) 881C* for examples of WBS.

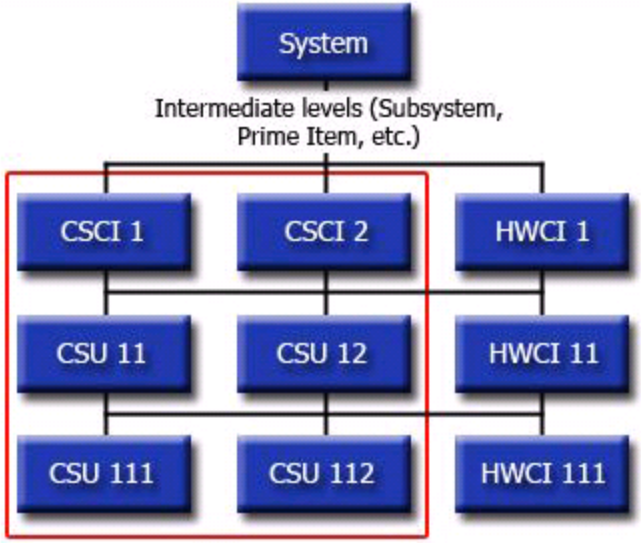
Step 1D: Develop Estimating Structure - Work Breakdown Structure (WBS)

The recently released MIL-STD-881C requires the use of a product-oriented WBS for all ACAT I, II, and III programs, and specifies updated WBSs by commodity in several appendices.

"A Work Breakdown Structure (WBS) provides a consistent and visible framework for defense materiel items and contracts within a program." The WBS is used in the software estimation steps that follow.

The WBS is an excellent tool for visualizing the software product, and the hardware WBS is often useful in developing the initial WBS for software.

The system is normally divided into hardware and software configuration items (HWCIs and CSCIs), with a CSCI or similar software module associated with each HWCI.



D

Long Description

The blocks in the WBS are arranged in the following manner:

- CSCI1, CSCI2 and HWCI1 top row
- CSU11, CSU12 and HWCI11 second row
- CSU111, CSU112 and HWCI111 third row

The description of each block type is as follows:

CSCI - Computer Software Configuration Item is an aggregation of software that satisfies an end use function and is designated for separate configuration management by the acquirer. CSCIs are selected based on tradeoffs among software function, size, host or target computers, developer, support concept, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors.

CSU - Computer Software Unit. An element in the design of a CSCI; for example, a major subdivision of a CSCI, a component of that subdivision, a class, object, module, function, routine, or database. Software units may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities.

HWCI - Hardware Configuration Item is an aggregation of hardware that satisfies an end use function and is designated for separate configuration management by the acquirer.

Step 1D: Develop Estimating Structure - Work Breakdown Structure (WBS), Cont.

Different projects may use different names for what are essentially CSCIs, such as modules, builds, or simply SIs, with CSCIs broken down further into CSCs and then CSUs, using the old MIL-STD-498 parlance.

A CSCI typically has no more than about 50,000 source lines of code (SLOC), and the corresponding WBS item encompasses all of the design, code, and unit test required to deliver the functionality allocated to that CSCI.

Within the top-level program WBS, both software content (CSCIs) and the associated activities (e.g., Integration and Test) should be identified.

The WBS need not be complex, nor does it need to be highly detailed. The WBS may evolve with the program, and the initial draft WBS should be established as soon as possible and should include all software associated with the program regardless of whether it is developed, furnished, or purchased.

Questions Managers Should Ask

Popup Text

Questions Managers Should Ask

Are the objectives of the estimate clear and correct?

- Are the tasks and activities included in (and excluded from) the estimate clearly identified and consistent with the objectives of the estimate?
- Is the content of the software estimate consistent with other program component estimate approaches and content?

Step 1D: Develop Estimating Structure - Work Breakdown Structure (WBS) - Electronic Systems (Appendix B)

C4ISR systems, including the software-defined radios of the ProRad program, would use the Electronic Systems WBS specified in Appendix B.

It is very flexible, allowing for any number of Prime Mission Product (PMP) elements, each with any number of associated subsystems.

A relevant excerpt is illustrated, with software elements highlighted in bold.

Software is inevitably involved in the integration and test items as well, including the element 1.5.3 Mock-ups / System Integration Labs (SILs).

1.1 Prime Mission Product (PMP) 1...n (Specify)
- 1.1.1 PMP Subsystem 1...n (Specify)
  - *1.1.1.1 PMP Subsystem Hardware 1...n*
  - ***1.1.1.2 PMP Subsystem Software Release 1...n***
  - *1.1.1.3 Subsystem Integration, Assembly, Test and Checkout*
- **1.1.2 PMP Software Release 1...n (Specify)**
  - ***1.1.2.1 Software Product Engineering***
  - ***1.1.2.2 Computer Software Configuration Item (CSCI) 1...n***
  - ***1.1.2.3 Subsystem Integration, Assembly, Test and Checkout***
- 1.1.3 PMP Integration, Assembly, Test and Checkout

1.2 Platform Integration, Assembly, Test and Checkout

Popup Text

Prime Mission Product (PMP)

The hardware and software used to accomplish the primary mission of the defense materiel item. This WBS element includes the design, development, and production of complete units (i.e., the prototype or operationally configured units, which satisfy the requirements of their applicable specifications, regardless of end use).

Step 1D: Develop Estimating Structure - Work Breakdown Structure (WBS) - Weapon System with Embedded SW

The illustration on the right, showing an excerpt from Appendix C Missile Systems, is typical of weapon systems with embedded software.

There may be software in each subsystem as shown in the level-4 and level-3 WBS elements such as 1.1.2.8 Propulsion Software and 1.3.8 Command and Launch Software, respectively. (Intervening lower-level WBS elements have been omitted here to keep the example to a manageable length.)

There is also a level-2 WBS element for system-level software, in this case 1.4 Missile System Software.

1.1	Air Vehicle
1.1.1	Airframe
1.1.2	Propulsion Subsystem (1…n) Specify
1.1.2.8	***Propulsion Software Release 1…n***
1.1.3	Power and Distribution
1.1.3.5	***Power and Distribution Software Release 1…n***
1.1.4	Guidance
1.1.4.4	***Guidance Software Release 1…n***
1.1.5	Navigation
1.1.5.3	***Navigation Software Release 1…n (Specify)***
1.1.6	Controls
1.1.6.6	***Controls Software Release 1…n***
1.1.7	Communications
1.1.7.3	***Communications Software Release 1…n***
1.1.8	Payload
1.1.8.5	***Payload Software Release 1…n***
1.1.15	Air Vehicle Software Release 1…n
1.2	Encasement Device
1.2.3	**Encasement Device Software Release 1…n**
1.3	Command and Launch
1.3.8	**Command and Launch Software Release 1…n**
1.4	**Missile System Software Release 1…n**

Step 1D: Develop Estimating Structure - Work Breakdown Structure (WBS) - AIS (Appendix K)

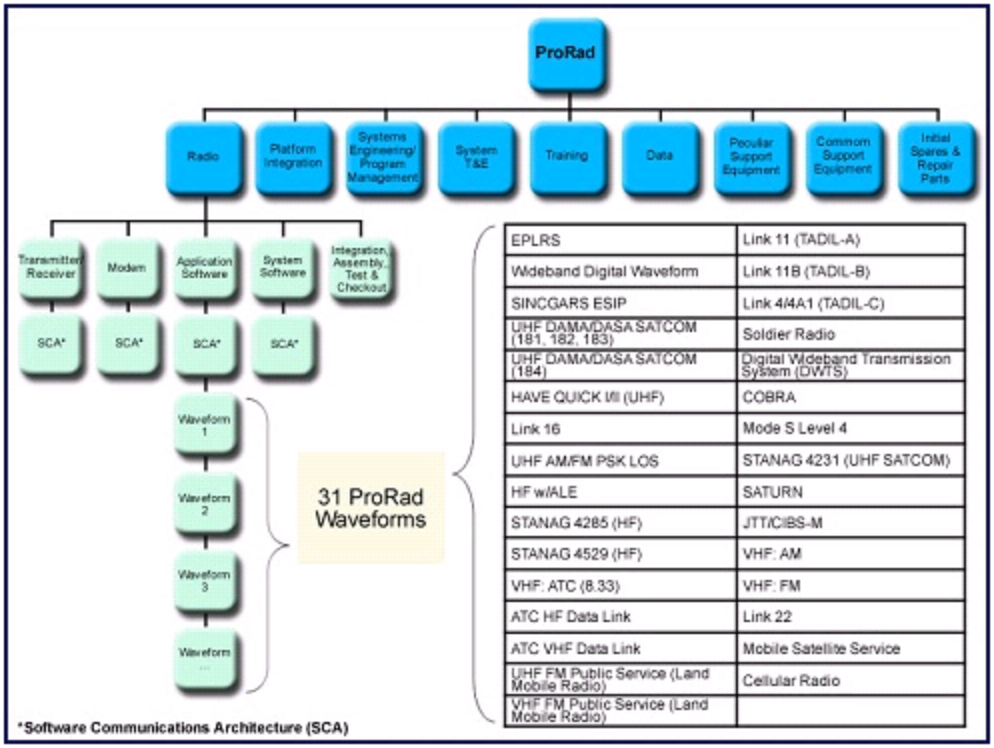
The illustration below is the main section of Appendix K for AIS. Similar to the flexibility allowed by Appendix B for Electronics, this WBS provides for multiple releases or increments.

Note the significant software content, not just in traditional developed software (1.1.1) but also software services (1.1.2), enterprise information systems such as ERP (1.1.3), and interfaces (1.1.4).

1.1	Automated Information System Prime Mission Product Release/Increment X
1.1.1	**Custom Application Software 1…n (Specify)**
1.1.1.1	*Subsystem Hardware*
1.1.1.2	***Subsystem Software CSCI 1…n (Specify)***
1.1.1.3	***Subsystem Software Integration, Assembly, Test and Checkout***
1.1.2	Enterprise Service Element 1…n (Specify)
1.1.2.1	*Enterprise Service Element Hardware*
1.1.2.2	***Enterprise Service Element Software CSCI 1…n (Specify)***
1.1.2.3	*Enterprise Service Element Integration, Assembly, Test and Checkout*
1.1.3	Enterprise Information System 1…n (Specify)
1.1.3.1	*Business Area Hardware*
1.1.3.2	***Business Area Software CSCI 1…n (Specify)***
1.1.3.3	*Business Area Integration, Assembly, Test and Checkout*
1.1.4	External System Interface Development 1…n (Specify)
1.1.4.1	*External System Interface Hardware*
1.1.4.2	***External System Interface Software CSCI 1…n (Specify)***
1.1.4.3	*External System Interface Integration, Assembly, Test and Checkout*

Step 1D: Develop Estimating Structure - Work Breakdown Structure (WBS) - ProRad WBS

Note that each of the 31 waveforms in the Program WBS represents a CSCI. Select the graphic below to view the enlarged Program WBS for the Joint ProRad project.



Step 1D: Develop Estimating Structure - Operating and Support (O&S) CES

OSD CAPE is tasked with establishing substantive guidance on the preparation and presentation of cost estimates. This responsibility encompasses the development of standard elements for the Research and Development (R&D), Investment, Operating and Support (O&S), and Disposal cost estimating categories.

A standard cost element structure (CES) promotes consistency in preparing and displaying estimates, and enables the CAPE to focus on high-cost/high-risk areas that have the greatest bearing on future costs.

The O&S CES for each weapon system category is designed to meet the needs of most CAPE reviews. However, the basic structure may have to be modified to accommodate the special features of some weapon systems. The CAPE must approve any changes to the standard CES structure before the costing work begins.

The key elements of interest in a software estimate are 6.4 Sustaining Engineering Support and 6.5 Software Maintenance Support, under 6.0 Sustaining Support.

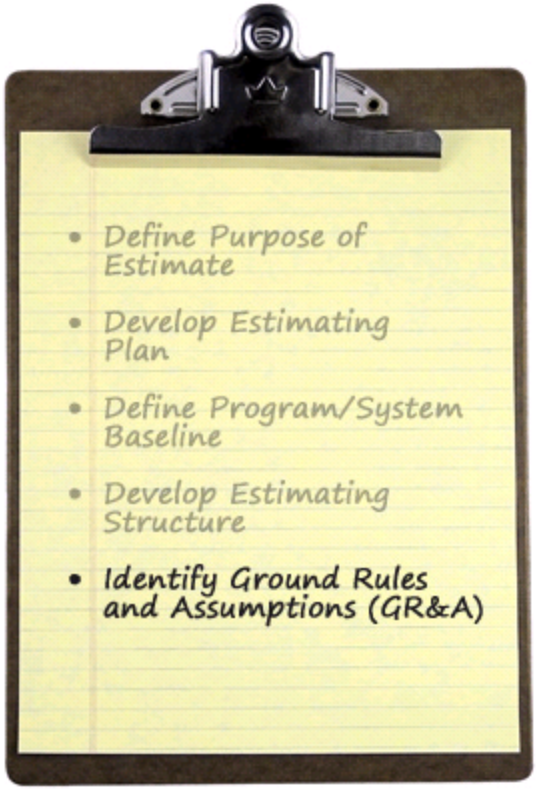
Step 1E: Identify Ground Rules and Assumptions (GR&A)

The fifth and final sub-step is to identify the Ground Rules and Assumptions (GR&A) associated with the estimate.

As much as possible, Assumptions (at least insofar as they impact cost) should reflect the best estimation of reality and should drive Ground Rules.

Often a practical distinction is not made between the two and they are simply referred to collectively. GR&A are important to bound and direct the effort associated with developing the estimate, starting with the collection and analysis of data.

They are also intended to help keep organizations developing independent estimates on the "same sheet of music." By clearly documenting GR&A ahead of time, the two estimating teams can avoid confusion and quibbles during the reconciliation process and instead focus on the substantive issues driving the estimates and the differences between them.



Questions Managers Should Ask

Popup Text

Ground Rules and Assumptions (GR&A)

Documented set of assertions upon which the fidelity of a cost estimate depends. As much as possible, Assumptions (at least insofar as they impact cost) should reflect the best estimation of reality and should drive Ground Rules.

Popup Text

Questions Managers Should Ask

- Have the factors that affect the estimate been identified and explained?
- Have assumptions been identified and explained?

Step 1E: Identify Ground Rules and Assumptions (GR&A) - Assumptions

Insofar as they represent assessments of future reality, assumptions should be backed up by program documentation (such as the CARD) and validated by independent subject matter experts (SMEs), so as to guard against unintentional excessive optimism.

The basic idea is to be careful not to "assume away" any costs. Such assumptions might include the degree of reuse to be achieved (which impacts Size), the skill of the available development staff (which impacts Capability), and required reliability (which impacts Complexity).

Other assumptions may be assumptions of convenience, which simplify the analysis but are not thought to significantly impact cost.

For example, if a portion of software development is to be subcontracted but the subcontractors performing the work are as yet unknown, an assumed productivity and labor rate might be used, though it should be representative of historical experience across the range of subcontractors who are likely to be available.

Popup Text

Subject Matter Expert (SME)

An individual asserted to be sufficiently expert in a subject to be able to give reliable testimony on the issue at hand. In risk analysis, SMEs are relied upon to predict the future growth in cost of a WBS element or program. SMEs are frequently the target of criticism by reviewers of SME-based risk methods, rendering such methods difficult to defend. Such criticism usually revolves around independence, sufficient expertise, or even the adequacy of such expert approaches. Nonetheless, they are an important risk analysis method, and can be the only recourse, particularly in systems or projects without precedent.

Step 1E: Identify Ground Rules and Assumptions (GR&A) - Ground Rules

Where possible, estimate ground rules should be explicitly linked to assumptions, preferably non-controversial ones, or other rationale. For example:

- It is assumed the program will pass its Milestone B review during FY12, therefore estimates should be reported in BY12$.
- The estimate will be used to justify the POM submission, therefore estimates should also be reported in TY$ using OSD inflation indices.
- The system is expected to have a service life of 20 years, therefore estimates should extend 20 years past full operational capability, referred to as "FOC+20."
- It is assumed that Company X will be the prime contractor, therefore their standard labor month of 160 hours will be used.
- The tech refresh cycle for COTS software is expected to be every three years, therefore maintenance agreements need to be priced for three years with an upgraded or replacement version to follow.
- It is assumed that only Level-1 and Level-2 defects will be addressed during sustainment, therefore software maintenance efforts should be estimated accordingly.

In this last case, do not make that assumption just to "get the cost down" if you think the real requirement is Levels 1 through 5.

For ProRad, a value of $16,000 per staff month was used as the loaded labor rate for estimating software development costs. This rate included all direct labor, vendor overheads (facilities/equipment), and other associated costs for program management and support engineering

Knowledge Review

Which of the following is least relevant in developing the estimating plan for a radar software estimate?

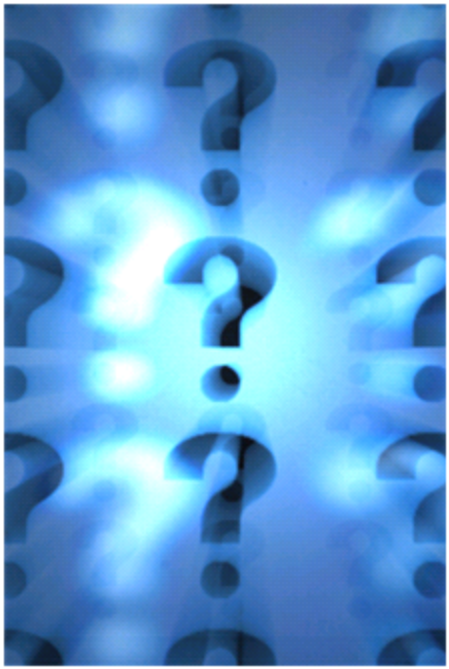
- [ ] The contractor used a particular off-the-shelf estimating model, but the program office does not have a license.
- [ ] Jasmine is leading the team because she has extensive experience in the estimation of radar software development.
- [x] Radar development is expected to take 36 months.
- [ ] Because a life-cycle cost estimate (LCCE) is called for, post-deployment software support (PDSS) must be included, even though those costs fall outside the Future Years Defense Program (FYDP).
- [ ] Software Integration and Test (I&T) should be included, but System I&T will be handled by a different estimating team.

Check Answer

Radar development is expected to take 36 months is least relevant. While the Timeline available for estimate development is an important consideration in the estimating plan, the time to do the estimate is not directly impacted by the development timeframe.

Knowledge Review

Classify missile flight control software, a base payroll system, and in-theater communications, respectively, as to type of software-intensive system.



- [ ] Embedded; C4ISR; AIS
- [x] Embedded; AIS; C4ISR
- [ ] C4ISR; Embedded; AIS
- [ ] C4ISR; AIS; Embedded
- [ ] AIS; C4ISR; Embedded

Check Answer

Embedded; AIS; C4ISR is the proper classification of missile flight control software, a base payroll system, and in-theater communications.

Knowledge Review

Which of the following is not a good assumption for a software estimate?

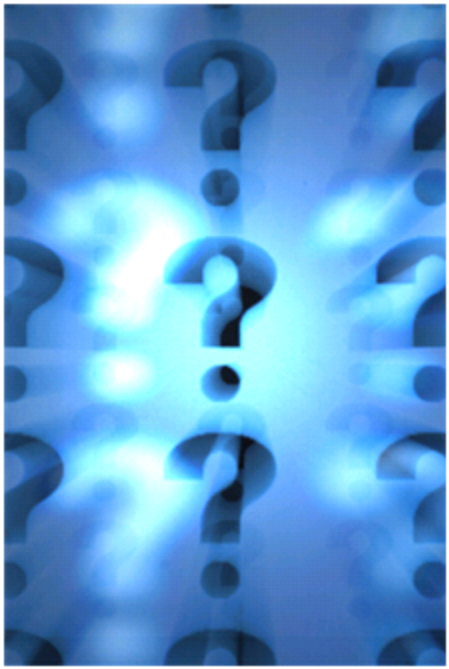
- [ ] Datalink functionality will be provided by A/N-XYZ-35.
- [ ] CSCI #1 will be written in C++ so as to take advantage of existing signal processing libraries.
- [x] CSCI #2 will be 25K SLOC and require 80 person-months of effort.
- [ ] The software will be written and tested on a LINUX platform.
- [ ] The development contractor will be Inordinate Complexity, Inc., since this project is expected to be a sole-source follow-on.

Check Answer

CSCI #2 will be 25K SLOC and require 80 person-months of effort is not a good assumption for a software estimate. Both sizing and effort need to be justified based on historical data and other inputs, not assumed.

Knowledge Review

Which of the following is not part of the MIL-STD-881C AIS WBS?



- [x] Operating System Software
- [ ] Enterprise Service Element Software
- [ ] Custom Application Software
- [ ] External System Interface Software
- [ ] Business Area Software

Check Answer

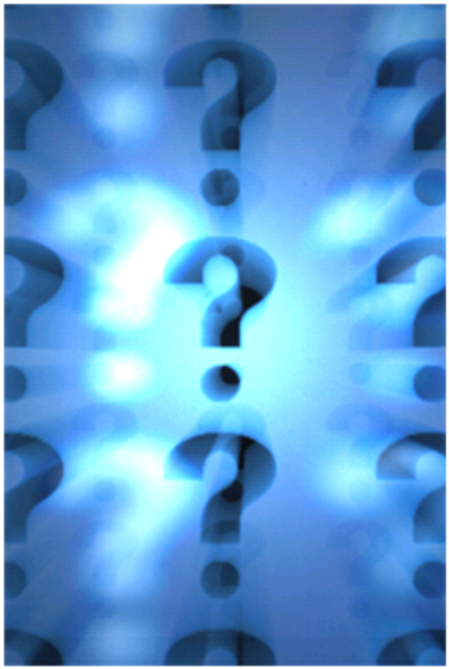
Operating System Software is not part of the MIL-STD-881C AIS WBS.

Knowledge Review

System Architecture should include:

- [ ] Hardware components only
- [ ] Software components only, including GFE and COTS
- [ ] Developed software components only
- [ ] Hardware components and developed software only
- [x] Hardware components and software components, including GFE and COTS

Check Answer



System Architecture should include **hardware components and software components, including GFE and COTS**. The architecture must show the components and associated functionality for the entire system.

Summary

This completes the Develop Scope and Approach lesson. In this lesson you learned:

- Software estimates support milestone reviews, budgets, and execution baselines.
- Software estimates require cost analysts familiar with software processes and available data and models.
- Software estimates require a clear description of the associated software-intensive system, such as provided by a Cost Analysis Requirements Description (CARD), including such critical information as software sizing and interfaces.
- Software estimates use a standard Cost Element Structure (CES), encompassing both a product-oriented Work Breakdown Structure (WBS) that delineates software from hardware and other items, and Operating and Support (O&S) cost elements such as software maintenance and sustaining engineering.
- Software estimates rely on ground rules and assumptions (GR&A), including reuse of developed software and functionality provided by Commercial Off-The-Shelf (COTS) software.

D

Long Description

Graphic illustrates the steps of the Cost Estimating process. The steps from left to right are:
Develop Scope and Approach (highlighted), Collect and Analyze Data, Develop Estimate Methodology, Consider Risk and Uncertainty, and Document and Present Estimate.

Lesson Completion

You have completed the content for this lesson.

To continue, select another lesson from the Table of Contents on the left.

If you have closed or hidden the Table of Contents, click the Show TOC button at the top in the Atlas navigation bar.